Departamento de Ciências e Tecnologias de Informação

Xprolog - Desenvolvimento de uma folha de cálculo dedutiva

Dissertação submetida como requisito parcial para obtenção do
Grau de Mestre em Informática e Gestão

Carlos Diogo Oliveira de Almeida

Orientador:
Prof. Doutor Pedro Nogueira Ramos

Co-Orientador:
Prof. Doutor Luís Miguel Botelho

Setembro, 2015

Xprolog - Desenvolvimento de uma folha de cálculo dedutiva

Resumo

As folhas de cálculo tradicionais apresentam limitações quando se trata de informação simbólica, que não é facilmente captada, principalmente quando se trata de raciocínio dedutivo. A presente dissertação tem como objetivo estender as capacidades das folhas de cálculo tradicionais, tal como o Excel, por exemplo, e desenvolver uma folha de cálculo com capacidades de representação de conhecimento e de raciocínio.

Desde a primeira folha de cálculo com capacidades dedutivas durante os anos 80 até ao presente foram surgindo várias ferramentas que visaram colmatar estas limitações, e no presente trabalho pretende-se dar continuidade a essa investigação e desenvolvimento de forma a dar mais um passo em busca de uma combinação cada vez mais perfeita entre as folhas tradicionais e a dedução lógica e representação de conhecimento.

Nesta dissertação é apresentado o Xprolog, uma ferramenta desenvolvida para Excel e que interpreta a linguagem Prolog, funcionando como um suplemento (*add-in*) para Excel. O Xprolog foi desenvolvido dentro da temática de folhas de cálculo dedutivas e, não sendo um sistema que possa neste momento ser considerado como um produto final, tem por objetivo dar um novo alento e promover um tema que tem vindo a perder algum fulgor nos últimos anos.

Palavras Chave: Folha de cálculo dedutiva, Excel, Prolog, Folha de cálculo tradicional, Excel *add-in*.

Abstract

Traditional spreadsheets have limitations when it comes to symbolic information, which is not easily captured, especially when it comes to deductive reasoning. This thesis aims to extend the capabilities of traditional spreadsheets such as Excel, for example, and develop one kind of spreadsheet with representation capabilities of knowledge and reasoning.

Since the first worksheet with deductive capabilities during the 80's up to the present, various tools have emerged that aimed to resolve these limitations, and in this work we intend to continue this research and development in order to take another search in step an increasingly perfect combination between traditional spreadsheets and logical deduction and knowledge representation.

In this thesis we introduce Xprolog, a tool developed for Excel that interprets the Prolog language, working as an add-in for Excel. The Xprolog was developed within the theme of deductive spreadsheets and not being yet, a system that can be considered as a final product, it aims to give new impetus and promote a theme that has been losing some enthusiasm in recent years.

Keywords: deductive spreadsheet, Excel, Prolog, traditional spreadsheet, Excel add-in.

Agradecimentos

Ao professor Pedro Ramos por todo o apoio e trabalho de orientação, pelas críticas construtivas, dicas, paciência, motivação, dedicação e esforço despendidos ao longo da realização desta dissertação.

Ao professor Luís Botelho pela co-orientação, pelas críticas construtivas e dicas de escrita, pelas revisões dos textos produzidos e paciência despendida ao longo da realização desta dissertação.

À minha família por todo apoio, encorajamento e motivação que foram fundamentais no realizar deste trabalho.

Aos meus amigos e colegas de trabalho e do curso que sempre me apoiaram e me motivaram a continuar até ao fim.

Obrigado a todos!

Glossário

Add-in: (plug-in ou add-on) é um módulo de extensão, ou seja, um programa de computador que adiciona funções a outros programas principais com funcionalidades específicas ou especiais.

Basic: (*Beginner's All-purpose Symbolic Instruction Code)* em português: Código de Instruções Simbólicas de Uso Geral para Principiantes, é uma linguagem de programação imperativa de alto nível que surgiu em 1964.

C: é uma linguagem de programação compilada de propósito geral criada em 1972. É uma das linguagens de programação mais populares.

CPPO: (cálculo de predicados de primeira ordem) é um sistema lógico que estende a lógica proposicional e que é estendida pela lógica de segunda ordem. As sentenças atómicas da lógica de primeira ordem têm o formato onde um predicado tem um ou mais argumentos.

CSP: *(constraint satisfaction problems)* são problemas matemáticos definidos como um conjunto de objetos onde o estado destes deve satisfazer um conjunto de restrições.

DARPA: *(Defense Advanced Research Projects Agency)* é a Agência de Projetos de Pesquisa Avançada de Defesa e foi criada em 1958 por militares e pesquisadores americanos.

Data Mining: é o processo de explorar grandes quantidades de dados procurando padrões consistentes.

Datalog: é uma linguagem de consulta baseada em Prolog e na lógica relacional. Nela o utilizador descreve as informações pretendidas sem fornecer um procedimento específico para obter tais informações.

LISP: é uma família de linguagens de programação criada em 1958 e projetada inicialmente para o processamento de dados simbólicos, é uma linguagem formal matemática. Tornou-se na principal linguagem da comunidade de inteligência artificial.

LPO: (Lógica de Primeira Ordem) sistema lógico que estende a lógica proposicional e que é estendida pela lógica de segunda ordem. As sentenças atómicas da lógica de primeira ordem têm o formato onde um predicado tem um ou mais argumentos.

MAC: Macintosh é o nome dos computadores pessoais fabricados e comercializados pela Apple Inc. desde 1984.

MS-DOS: *(MicroSoft Disk Operating System)* é um sistema operacional, adquirido pela Microsoft para ser usado na linha de computadores IBM PC.

OWL: *(Web Ontology Language)* é uma linguagem de descrição de ontologias especialmente pensada para a Web. É a linguagem de descrição de ontologias mais disseminada nos dias de hoje.

Prolog: é uma linguagem de programação enquadrada no paradigma da programação em lógica matemática e especialmente associada com a inteligência artificial.

RETE: algoritmo de correspondência de padrões para implementação de sistemas de regras de produção.

SWRL: *(Semantic Web Rule Language)* é uma proposta de linguagem para a web semântica que pode ser usada para expressar regras, mas também lógica.

VB: é uma linguagem de programação integrante do pacote *Microsoft Visual Studio* (produzido pela Microsoft). É uma aperfeiçoamento do BASIC e dirigida por eventos, possuindo uma ambiente de desenvolvimento integrado totalmente gráfico.

VBA: é uma implementação do Visual Basic incorporada em todos os programas do *Microsoft Office*. Substitui e estende as capacidades das linguagens de programação em macros e pode ser usada para controlar quase a totalidade dos aspetos da aplicação anfitriã.

XML: *(eXtensible Markup Language)* é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo I - Introdução

1.1.Enquadramento

A folha de cálculo é uma tabela de duas dimensões (composta por linhas e colunas) onde se podem realizar cálculos e apresentar vários tipos de dados. A utilização da folha de cálculo é hoje em dia, tal como desde já há longos anos, bastante comum e popular entre uma variedade enorme de pessoas e que se estende a estudantes, professores, investigadores, executivos, etc. O seu uso tende a cada dia que passa a aumentar ainda mais e esta popularidade deve-se sobretudo à sua grande usabilidade e utilidade, podendo-se considerar uma ferramenta bastante acessível e de aprendizagem de utilização fácil e rápida por parte do utilizador. A sua grande usabilidade comprova-se quando se verifica que um utilizador inexperiente pode, com apenas algumas instruções e cliques de rato, criar e manipular grandes quantidades de dados e automatizar fácil e rapidamente cálculos bastante complexos. A folha de cálculo é também, a nível estratégico e de tomada de decisão, bastante importante nos dias que correm (Cervesato, 2005), permitindo criar representações gráficas que sintetizam e apresentam a informação de forma clara e intuitiva. Contudo uma grande parte das decisões que se tomam não são numéricas, ou mesmo a grande maioria não são baseadas apenas em números mas necessitam de raciocínio dedutivo como é demonstrado mais adiante neste trabalho.

Desde a sua introdução, em termos informáticos, no final da década de 70, até aos dias de hoje o conceito da folha de cálculo não foi sofrendo alterações significativas, mas foi apresentando constantes melhorias, quer de usabilidade quer de novas funcionalidades, surgindo na década de 80 o conceito de folha de cálculo dedutiva também conhecidas por DSS (*Deductive Spreadsheet*) (C. R. Ramakrishnan et al., 2007) .

As folhas de cálculo tradicionais são essencialmente folhas com objetivo de computação aritmética e visualização gráfica, ou seja, não estão desenhadas, tradicionalmente para suportar raciocínio dedutivo. Embora as computações em lógica booleana sejam possíveis, as folhas de cálculo mais disseminadas (por exemplo, MS Excel, Kingsoft Spreadsheets, LibreOffice Calc, OpenOffice.org Calc, entre outros) não efetuam outros tipos mais avançados de raciocínio lógico, por exemplo dedução e recursividade.

1.2.Assunto e Motivação

Como referido acima no enquadramento, em situações de tomada de decisão, por exemplo, podemos tomar um caso de dedução lógica onde temos uma companhia aérea com voos diretos desde Porto a Lisboa e desde Lisboa a Istambul. Quando um cliente que vai voar do Porto e tem como destino Istambul procura o voo respetivo, não tem ligação direta. Mas indiretamente tem a possibilidade de voar desde o Porto até Lisboa e desde Lisboa até Istambul. Este tipo de dedução recursiva não é possível numa folha de cálculo tradicional, imaginando esta situação aplicada em Excel percebemos que este tipo de informação não numérica, ou seja, simbólica não é facilmente captada por uma folha de cálculo tradicional.

Apesar das folhas de cálculo oferecerem alguns operadores lógicos mais simples (da álgebra de Boole), estão apenas aptos a ser usados em simples testes condicionais e não em raciocínio dedutivo. As aplicações informáticas que suportam raciocínio dedutivo, como linguagens de programação lógica ou algumas bases de dados exigem um nível de sofisticação elevado e que mesmo muitos programadores não têm.

Com o intuito de aproveitar a grande capacidade e sucesso das folhas de cálculo tradicionais e aperfeiçoar as suas capacidades de raciocínio dedutivo, propomos e desenvolvemos uma extensão ao Excel capaz de raciocínio relacional e dedutivo, uma folha de cálculo com capacidades de representação de conhecimento e de raciocínio. Para simplificar refere-se durante grande parte do trabalho a estas folhas como folhas dedutivas, apesar de algumas das suas funcionalidades poderem não envolver dedução.

A folha de cálculo dedutiva desenvolvida poderá servir como um motor impulsionador de uma tendência já existente desde os anos 80, de desenvolvimento de folhas de cálculo dedutivas aproveitando a grande disseminação, usabilidade e utilidade das folhas tradicionais e criar uma ferramenta com maior potencialidade não só para uso em Inteligência Artificial, mas sobretudo para situações mais comuns enfrentadas por executivos, alunos e professores, entre outros, como também será demonstrado mais a frente na dissertação. Tornar problemas complexos e por vezes praticamente impossíveis de resolver numa folha de cálculo tradicional em tarefas bastante mais

simples para o utilizador, através da utilização de uma tecnologia mais adequada é a principal motivação desta dissertação.

Sendo também este um tema ainda com pouco desenvolvimento e que no nosso entender, não tem tido, nos últimos anos, a atenção devida, coloca-nos um duplo desafio de procurar desenvolver algo que traga claras vantagens para os diferentes tipos de utilizadores das folhas de cálculo, mas também o de disseminar tecnologias e abordagens oriundas da inteligência artificial, como o raciocínio dedutivo e a programação em lógica, numa comunidade muito alargada, como a comunidade de utilizadores das folhas de cálculo.

1.3.Objetivo

O objetivo central do trabalho é o de desenvolver o conceito de folha de cálculo com capacidades relacionais e dedutivas, desenvolver uma folha de cálculo com capacidades de representação de conhecimento e de raciocínio, permitindo assim a resolução de problemas comuns dos utilizadores de folhas de cálculo tradicionais.

O SWI-Prolog foi escolhido por se tratar de uma implementação em código aberto da linguagem de programação Prolog, mantida por Jan Wielemaker da Universidade Vrije em Amsterdão, largamente compatível com a norma ISO-Prolog (Wielemaker, 1999), e que tem uma vasta comunidade de utilizadores desde o seu lançamento em 1987.

A ferramenta ideal será aquela onde o utilizador final consegue abrir uma folha de Excel, carregar uma base de dados, definir relações, restrições e regras sobre esses dados e executar uma interrogação que exige raciocínio dedutivo, e instantaneamente obter uma resposta, que pode inclusivamente reaproveitar como nova base de dados.

1.4.Estrutura da Dissertação

Esta dissertação inicia-se com a introdução do tema a desenvolver, no capítulo I, o qual apresenta o enquadramento do assunto. De seguida, a motivação do desenvolvimento da ferramenta construida e enquadramento prático com a definição e explicação dos objetivos que se procuraram alcançar.

No capítulo II apresenta-se a definição e descrição da linguagem Prolog, seguindo-se uma explicação de funcionamento das folhas de cálculo tradicionais. Seguidamente é apresentada uma evolução histórica das folhas de cálculo dedutivas, desde os seus primórdios até aos dias de hoje, culminando com uma tabela matriz de comparação das folhas analisadas, organizada por sete características definidas pelo autor da dissertação.

No capítulo III, é apresentada a ferramenta XProlog que foi desenvolvida. É feita uma abordagem técnica e referidas as principais características da ferramenta, assim como as suas limitações.

Posteriormente são dados exemplos que evidenciam as vantagens da utilização da ferramenta. É ainda analisada a ferramenta criada, no quadro das sete características referidas no capítulo II.

No capítulo IV, é analisada com recurso a exemplos, a utilização de suplementos de Excel para realizar tarefas ilustrativas, e depois aplicando a ferramenta XProlog e analisados os resultados. As ferramentas que se usaram foram o PowerQuery e o PowerPivot que são dois dos suplementos para Excel mais usados nos dias de hoje.

No capítulo V é apresentada a conclusão do trabalho realizado, com a descrição de potenciais desenvolvimentos futuros no XProlog e considerações finais. O trabalho é finalizado com a bibliografia apresentada no capítulo VI.

Capítulo II - Estado da Arte

No estado da arte, foram analisadas as principais características inerentes à ferramenta desenvolvida. Surge aqui uma breve explicação da programação lógica e da linguagem utilizada (Prolog). É analisada a evolução das folhas de cálculo tradicionais até ao surgimento das folhas de cálculo dedutivas.

"O pesquisador, ao desenvolver para o leitor o assunto, deixa de ser por um momento investigador, para se tornar o filósofo de seu trabalho. Abandona as técnicas da pesquisa com que já se habituara, para usar os recursos da lógica da demonstração" (Salomon, 1991).

2.1.Programação Lógica

A programação em lógica teve as suas origens no cálculo de predicados (Frege, 1879) e é um paradigma de programação que faz uso da lógica matemática. John McCarthy em 1958 foi o primeiro a publicar uma proposta de uso da lógica matemática em programação.
A primeira linguagem de programação lógica criada foi a Planner (Xavier, 2007) que é uma linguagem orientada a padrões de planos procedimentais de asserções e objetivos. Pouco depois surgiu a linguagem Prolog que foi desenvolvida como uma forma de simplificação do Planner.
A primeira implementação da linguagem Prolog foi realizada na Universidade de Aix-Marseille em 1972 por Alain Colmerauer. Os programas em lógica apresentam características que os diferenciam de programas convencionais, tais como o facto de poderem ser usados para expressar conhecimento independentemente da implementação, o que torna os programas mais flexíveis e compreensíveis. Além disso apresentam boa redigibilidade, sendo fáceis de aprender, permitem reusabilidade (Casanova, 19987). Estes programas permitem uma separação entre o conhecimento e utilização, pois a arquitetura da máquina pode ser alterada sem alterar os programas e o seu código subjacente. Outras características importantes são o facto de permitirem retrocesso (backtracking) e de poderem ser usados em disciplinas não computacionais (Hogger, 1996).

O Prolog é uma linguagem de programação mais simbólica do que numérica, é uma linguagem simples, mas poderosa, fundamentada na lógica simbólica (Bratko, 2001). O processamento simbólico é muito mais preciso que o processamento numérico, pois as operações com valores numéricos estão sujeitos a erros de arredondamento que se vão acumulando em operações sucessivas, enquanto que as operações simbólicas não geram esses erros.
Além destas características, os programas em lógica são de fácil modificação, e têm em conta todas as soluções possíveis (Casanova, 1987).

2.2.Prolog

O Prolog é uma linguagem de programação enquadrada no paradigma da programação em lógica matemática e é baseada num subconjunto de lógica de primeira ordem, as cláusulas de Horn (Alfred Horn, 1951). Em Prolog, a tarefa do programador é a de especificar o problema que deve ser solucionado, e daí a razão das linguagens de programação em lógica poderem ser vistas tanto como linguagens para especificação formal, como de programação em computadores.
O Prolog apresenta estruturas de controlo e conhecimento separadas, onde a componente lógica corresponde à definição do que deve ser solucionado e o componente de controlo estabelece como a solução pode ser obtida O programador apenas necessita de definir a componente lógica de um algoritmo e o controlo da execução é exercido pelo sistema de programação em lógica utilizado (Palazzo, 1997). Assim sendo, para problemas que exijam que se exerça controlo diferente daquele que é oferecido implicitamente no executor de Prolog, a linguagem dispõe também de mecanismos explícitos de controlo que podem ser usados pelo programador.
Ao contrário de linguagens como C, por exemplo, um programa em Prolog, em termos declarativos, não é descrito como um procedimento para se obter a solução de um problema, pois o sistema utilizado no processamento de programas em lógica executa o seu próprio procedimento a adotar na sua execução (Oliveira et al., 2013). Ao mesmo tempo, o Prolog pode ser visto também como uma base de dados, mas com maior abrangência, pois permite a representação de regras, a que se chama geralmente uma base de conhecimento (Covington et al., 1997). Assim em vez de ter uma tradicional base de dados onde podemos ter por exemplo,

"Maria é do sexo feminino", podemos com o Prolog definir, "todas as mulheres são do sexo feminino".

Um programa Prolog declarativo é assim uma base de conhecimento composta por regras e fatos. Quando é especificado um problema ou uma pergunta ao sistema, o sistema usa um algoritmo de busca que usa a base de conhecimento para resolver o problema ou responder à questão. É uma linguagem de uso geral que é especialmente associada com a inteligência artificial e *data mining* (Palazzo, 1997).

Assim, algumas das mais importantes propriedades da linguagem Prolog são o fato de ao mesmo tempo que são uma linguagem de programação são também uma linguagem de especificação, apresenta capacidade dedutiva, funciona de forma não determinística. Além disso, permite representação de relações revertíveis onde os programas não distinguem entre os argumentos de entrada e os de saída, tem interpretação declarativa e procedimental e é naturalmente recursivo, suportando código recursivo e iterativo na descrição de processos e problemas (Palazzo, 1997).

Considerando, por exemplo, uma árvore genealógica onde temos os membros da família que são os objetos, e onde é possível definir uma relação entre eles. Neste caso a relação denominada "progenitor" vai associar um objeto "Rui", por exemplo, a outro objeto "João" onde um é o progenitor do outro. Uma relação como progenitor é facilmente definida em Prolog estabelecendo os tuplos de objetos que satisfazem esta relação. Os argumentos das relações podem ser objetos concretos ou genéricos, sendo os primeiros os números e os átomos (rui, maria, ...) e os segundos as variáveis tais como (X,Y,Z,...). além disto, um programa Prolog é ainda composto por cláusulas que são encerradas por um ponto "." (Palazzo, 1997).

Exemplo 1:

```
progenitor(rui, maria).
```

O exemplo acima lê-se: "Rui é progenitor de Maria"

Exemplo 2:

```
homem(rui).
```

Quando usamos apenas um objeto, o predicado passa a ser uma característica do próprio objeto.

Exemplo 3:

```
progenitor(rui, maria).
progenitor(joana, maria).
progenitor(rui, joao).
progenitor(joao, ivo).
progenitor(maria, jose).
progenitor(antonio, jose).
```

Este programa é composto por seis cláusulas onde cada uma especifica um fato sobre a relação progenitor.

Assim, em Prolog, poderíamos fazer a consulta, por exemplo:

Maria é progenitor de José ?

O sistema irá responder afirmativamente. Diz-se que haveria sucesso.

Assim como se fosse feita a consulta:

Maria é progenitor de Rui?

Haveria insucesso.

Num sistema Prolog a questão seria colocada desta forma:

```
?- progenitor(maria, jose).
true
```

Segue abaixo a figura 1 que é uma representação gráfica do programa do Exemplo 3.

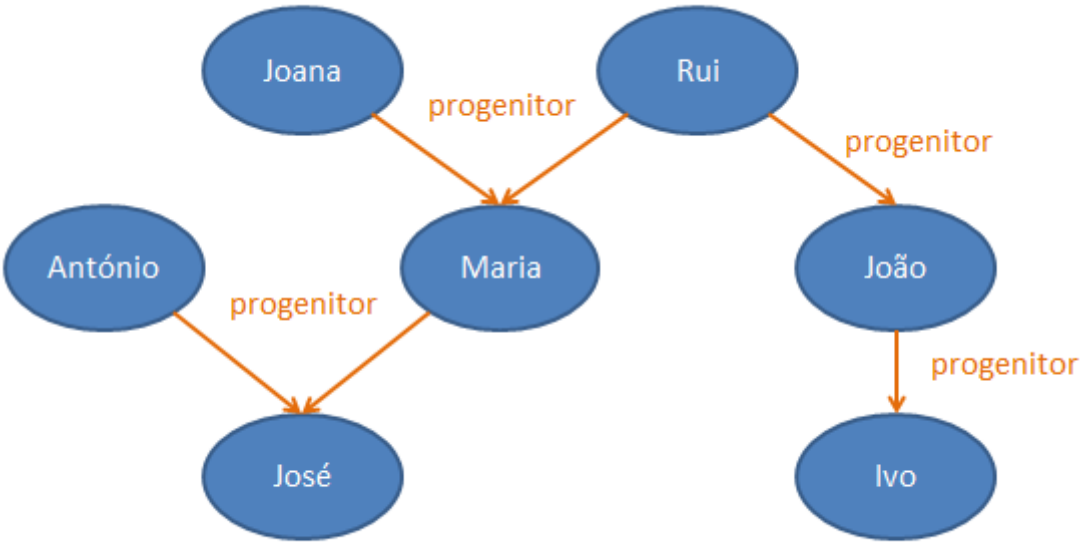


Figura 1 - Árvore genealógica.

O sistema permite também que se façam perguntas mais profundas, tal como: Quem é o progenitor Ivo?

Para este caso introduz-se uma variável X como argumento correspondente ao progenitor de Ivo, e o sistema não vai responder True ou False, mas vai procurar um valor de X na base de dados e responder:

Exemplo 4:

?- progenitor(X, ivo).

X= joao

Também poderia ser feita a questão "Quem são os filhos de Rui?" e coloca-se neste caso uma variável na posição do argumento correspondente aos filhos de Rui.

?- progenitor(rui, Y).

Y=joao;

Y=maria;

False

Questões mais complexas podem ser formuladas, tal como "Quem são os avós de José?"
Neste caso, como no programa definido no exemplo 3 não existe a relação avós, a consulta pode ser dividida em duas partes para obter a resposta certa:

1.Quem é o progenitor de José?
2.Quem é o progenitor do progenitor de José?

A consulta é escrita como sequência de duas consultas simples: "encontrar X e Y tais que X é progenitor de Y e Y é progenitor de José"

Exemplo 5:
?-progenitor(X, Y), progenitor(Y, josé).
X=rui Y=maria

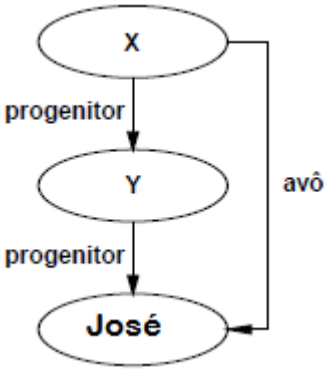


Figura 2 - A relação avô em função da relação progenitor.

De modo similar pode ser feita a consulta: "Quem é neto de Rui?":
Exemplo 6:
?-progenitor(rui, X), progenitor(X, Y).
X=maria Y=jose;
X=joao Y=ivo.

Definindo a relação filho de modo semelhante à utilizada para definir a relação progenitor.

Exemplo 7:
filho(maria, rui).

Pode-se agora fazer uso do facto das relações progenitor e filho serem inversas para fazer a seguinte declaração lógica:

Para todo X e Y
Y é filho de X se
X é progenitor de Y.

A cláusula em Prolog fica:

filho(Y, X) :- progenitor (X, Y).

Que se lê: Para todo X e Y, se X é progenitor de Y, então Y é filho de X.
As cláusulas em Prolog são chamadas de regras ou factos. As regras especificam que algo pode ser verdadeiro se algumas condições forem satisfeitas.
A conclusão das regras é a parte esquerda da cláusula e a parte da direita da cláusula é a condição.
O símbolo ":-" traduz-se por "se". Este separa a conclusão da conclusão (ou cabeça da cláusula) da condição ou corpo da cláusula.

A especificação da relação mãe entre dois objetos pode ser descrita na seguinte declaração:

Para todo X e Y
X é mãe de Y se
X é progenitor de Y e
X é feminino.

em Prolog:
mãe(X,Y) :- progenitor(X,Y), feminino(X).

A vírgula entre as duas condições indica a conjunção.

No Prolog pode ser também utilizado um estilo recursivo.
Se pensarmos por exemplo na relação antepassado, a ideia passa por definir a relação em termos de si mesma, usando a recursividade:

Para todo X e Z
X é antepassado de Z se
existe um Y tal que
X é progenitor de Y e
Y é antepassado de Z.

A cláusula Prolog correspondente é:

```
antepassado(X, Z) :-
        progenitor(X, Z).
antepassado(X, Z) :-
        progenitor(X, Y), antepassado(Y, Z).
```

Assim temos uma programa que demonstra a relação antepassado que é composto por duas regras, para os antepassados diretos e para os antepassados indiretos.

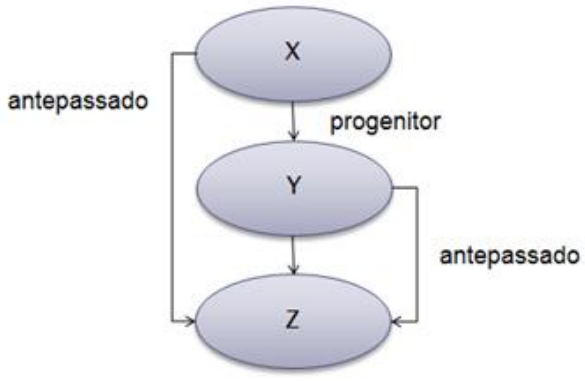


Figura 3 - Formulação recursiva da relação antepassado.

No Prolog existem ainda os operadores relacionais e os aritméticos.

Os operadores relacionais são:

Igualdade: =:= **ou** ==

Diferença: \== **ou** <> Menor que: <

Maior que: >

Menor ou igual: =<

Maior ou igual: >=

Os operadores aritméticos são:

Adição: +

Subtração: -

Multiplicação: ***

Divisão: /

Divisão inteira: **div**

Resto da divisão inteira: **mod**

Potenciação: ^

2.3.Folhas de Cálculo Tradicionais

Neste ponto do capítulo 2, é apresentada a evolução das folhas de cálculo tradicionais e a sua grande importância nos dias de hoje, assim como as suas principais características, que serão importantes de reter de forma a se perceber as melhorias apresentadas no Xprolog.

"As folhas de cálculo tornam fáceis cálculos que normalmente seriam complexos. E tornam ainda mais fácil fazê-lo incorretamente"
(Richard Scoville, 1999).

Relativamente a folhas de cálculo tradicionais, a primeira de que há registo é o VisiCalc, que foi criada por Dan Brickin e Bob Frankston em 1979 (Grad, B. 2007). O VisiCalc ganhou popularidade como aplicação da Apple e foi posteriormente vendida à Lotus Development Corporation levando ao desenvolvimento do Lotus 1-2-3 em 1983, que foi uma das primeiras folhas de cálculo disponíveis para o mercado (Power, 2004).

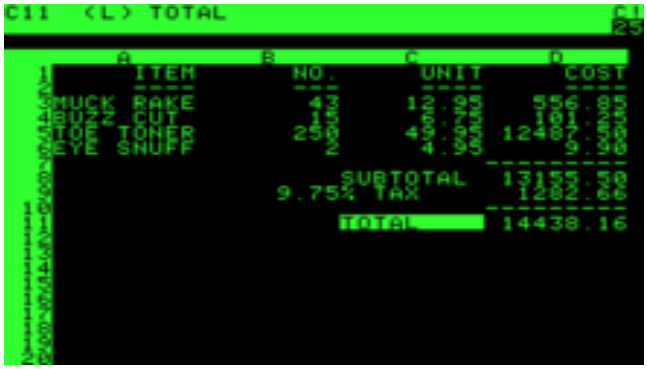


Figura 4 - Visicalc a correr no sistema Apple II.

Na figura 4 podemos verificar o aspeto do Visicalc e na figura 5 o Lotus 1-2-3.

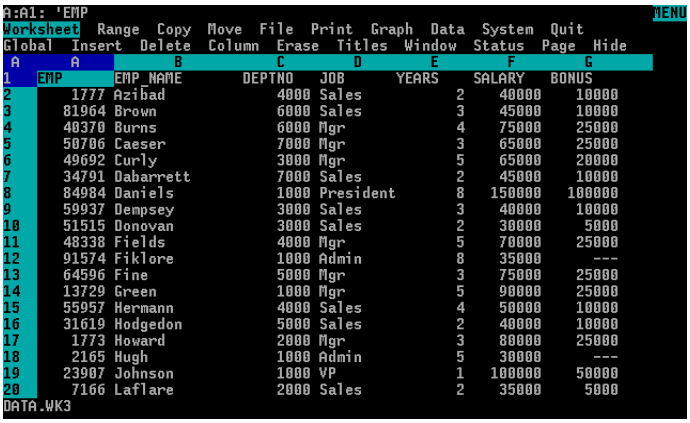


Figura 5 - Lotus 1-2-3.

Em 1982 a Microsoft tinha desenvolvido uma folha de cálculo chamada Multiplan, que era bastante popular em sistemas CP/M, mas perdia em popularidade para o Lotus 1-2-3 em sistemas MS-DOS.

Figura 6 - Multiplan.

Esta desvantagem levou a Microsoft a desenvolver um novo programa. Surgia em 1985 a primeira versão do Excel lançada para MAC, surgindo em 1987 a primeira versão para Windows. A Lotus demorou bastante para trazer o Lotus 1-2-3 para o Windows e em 1988 o Excel ultrapassa o 1-2-3 em vendas (Power, 2004).

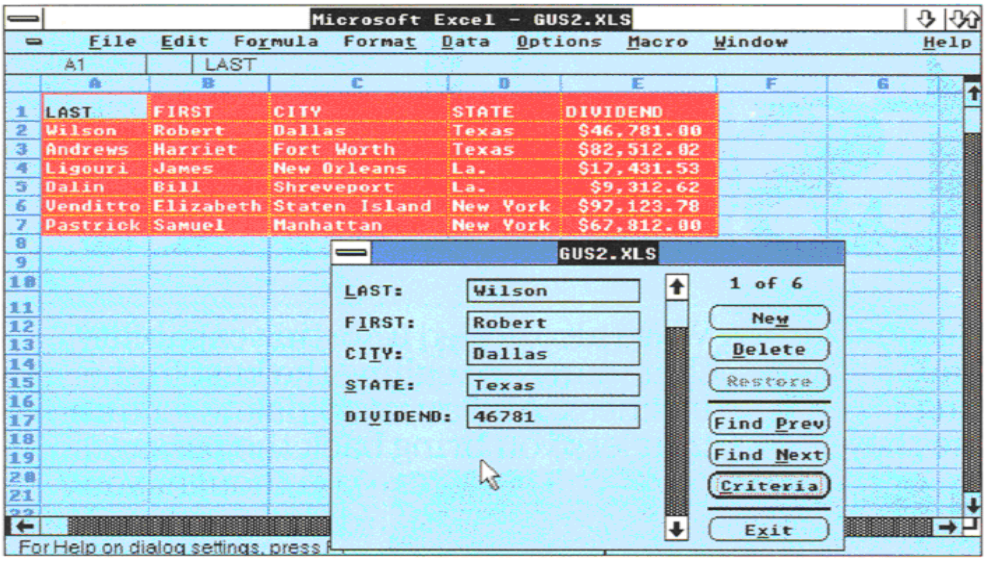


Figura 7 - Microsoft Excel 2.0 (1987).

A Microsoft foi através dos anos aumentando a sua vantagem com lançamento regular de novas versões, sendo a mais atual para Windows o Excel 15 ou mais conhecido como Microsoft Excel 2013 (Power, 2004).

As folhas de cálculo tradicionais têm a função de permitir inserir, organizar e analisar informação. O utilizador tem a possibilidade de inserir dados, guardá-los, processá-los e de os analisar, quer estes sejam texto, números ou mesmo fórmulas. Os elementos básicos de uma folha de cálculo são as células, e são identificados pela interseção de uma linha e de uma coluna. Na célula podem ser colocados valores, referências, formatos e fórmulas.

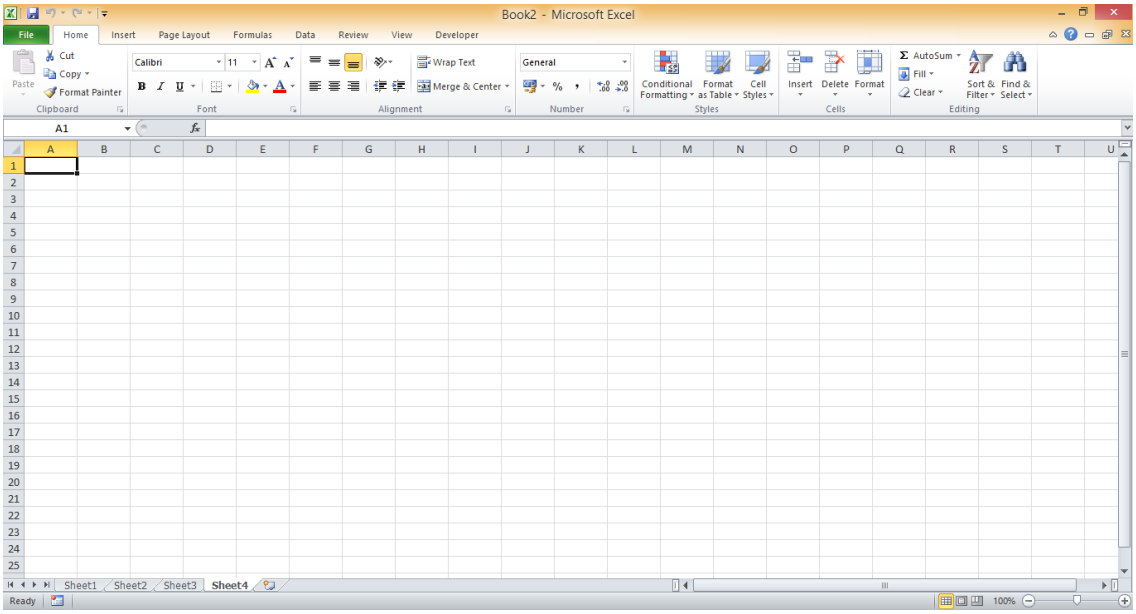


Figura 8 - Ambiente de uma folha de cálculo tradicional (Excel 2010).

A figura 8 representa uma folha de cálculo do Excel 2010, sobre a qual foi desenvolvido o Xprolog.

2.4.Folhas de Cálculo Dedutivas

As folhas de cálculo dedutivas, tal como referido na introdução, surgem com intuito de juntar a usabilidade e características de cálculo das folhas tradicionais com o raciocínio dedutivo.

Algumas das vantagens de uma folha de cálculo dedutiva são as de permitir a relação entre as células de muitos para muitos, além de permitir que a propagação das células se faça de forma multi-direcional. Outra das vantagens de uma folha de cálculo dedutiva passa por permitir criar fórmulas que envolvam inequações por exemplo; além de permitir raciocínio dedutivo. Seguidamente é apresentada a evolução deste tipo de folhas de cálculo desde os seus primórdios até ao seu estado mais atual (Kassof et al., 2007).

2.4.1.Evolução folhas de cálculo dedutivas

A primeira folha de cálculo dedutiva da qual se tem conhecimento é a LogiCalc (Kriwaczek, 1988) nos inícios dos anos 80 e que posteriormente fez parte da tese de Mestrado de Kriwaczek

no Imperial College em Londres em 1988. As fórmulas a aplicar nas células da folha de cálculo são escritas usando a sintaxe da folha de cálculo tradicional, mas depois são traduzidas internamente para regras de Prolog. Por exemplo, se fosse definido que A1 = A2 + A3, a LogiCalc internamente convertia para o seguinte em Prolog.

definido como (‘A1’, X)
igual a(‘A2’, Y) [será igual_a(‘A2’, Y)?]
igual a(‘A3’, Z)
e X = Y + Z
Esta ferramenta permite que áreas da folha de cálculo sejam guardadas como relações de base de conhecimento e depois são feitas consultas sobre elas. Uma das limitações desta abordagem é que as consultas apenas podem apresentar conjunções.

Exemplo:

```
masculino(jose).
masculino(andre).
feminino(maria).
progenitor(jose,andre).
progenitor(jose,maria).

filho(X,Y):-progenitor(X,Y),masculino(Y).
filha(K,Z):-progenitor(K,Z),feminino(Z).
```

Ou seja, este tipo de base de conhecimento pode ser guardado na folha de cálculo e podem ser feitas *queries* conjuntivas:

Qual [(X,Y):-progenitor(X,Y) e masculino(Y)]
Lê-se: Qual a combinação X e Y tais que X é progenitor de Y e Y é masculino?

Outra limitação são os tuplos (linhas) que são carregados numa única célula, em vez de distribuídos por colunas.

Exemplo:
% pessoa(nome, sexo, idade)
pessoa(jose, masculino, 35).

Numa situação como o exemplo acima, a LogiCalc colocaria a informação da pessoa numa única célula, por exemplo B2, em vez de distribuir por três colunas, por exemplo B2, C2 e D2 como se pode verificar na figura 9.

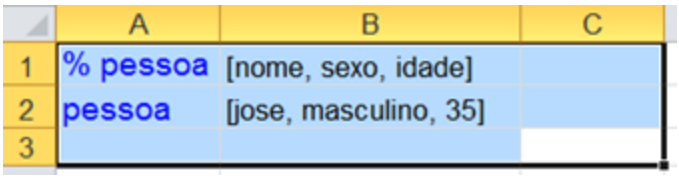


Figura 9 - Exemplo em Excel.

Posteriormente é desenvolvido um complemento ao LogiCalc: um sistema com interface de folha de cálculo para programas lógicos (van Emden et al. 1986). A capacidade de restrição interativa apresentada na LogiCalc é melhorada no sistema de van Emden, que consiste numa ferramenta para *queries* incrementais integrada na matriz da folha de cálculo. Com essa ferramenta, o sistema permite que o cálculo anterior seja reutilizado quando se adiciona uma nova restrição.
Van Emden acreditava que as folhas de cálculo existentes na altura deixavam muito potencial por explorar. Os autores (van Emden et al. 1986) referem que, ao contrário de linguagens como Basic, o interface das folhas de cálculo não poderia ser feito da mesma forma direta com o Prolog. O mecanismo de consulta habitual de Prolog não fornece o mesmo tipo de interação que uma folha de cálculo requer, mas poderia ser fornecida por consulta incremental, ou seja, um novo mecanismo de consulta para Prolog.
No geral, e tal como referido anteriormente sobre a LogiCalc, as restrições permitidas no sistema de van Emden são também bastante simples quando comparadas com todo o potencial do Prolog, pois consistem apenas de uma única conjunção de condições que pode ser expresso por uma

única regra de Prolog. Ou seja, estes sistemas não têm em conta que o Prolog permite que um conjunto de regras seja definido (por exemplo, condições disjuntivas) e além disso não têm em conta que as regras podem ser recursivas.

Ainda em 1986 foi desenvolvido o PERPLEX (Spenke & Beiken 1986), que permite que novas regras sejam definidas em função de anteriores, usando regras de Prolog. Os autores referem que o PERPLEX foi implementado com sucesso numa máquina simbólica LISP. Assim como os seus antecessores, o PERPLEX permite a criação de regras usando as relações das bases de dados. Além disso, permite restrições com base em predicados internos, e portanto permite operações aritméticas, concatenação e afins. Cada predicado interno (*built in*) tem um conjunto de input-*output* que identifica cada argumento do predicado aceitando um input ou produzindo um *output*.

Por exemplo:
o predicado com três argumentos tais como (A1 A2 A3) significando por exemplo, que A1+A2=A3, tem 4 modos *input-output:*
(in in out)
(in out in)
(out in in)
(in in in)

Para predicados definidos pelo utilizador, de acordo com as regras Prolog, o conjunto de modos *input-output* é calculado automaticamente. A forma do PERPLEX lidar com restrições numéricas é bastante simples mas incompleta, pois por exemplo, considerando a restrição +(A1 A1 10). Embora A1 tenha de ser igual a 5, o PERPLEX não consegue determinar o seu valor. Algo mais sofisticado seria necessário para resolver os problemas dos modos de entrada e saída para resolver tal limitação; tendo em conta que adicionando o modo *input-output* (out out in) iria acionar propagação quando +(A1 B1 2) é dado, o qual não é resolvido. Considerando por exemplo as restrições, C3 >= 4 e C3 <= 4. Neste caso C3 = 4, mas as restrições devem ser consideradas juntas para o determinar, e o PERPLEX não o consegue fazer também.

Mais tarde é desenvolvida a Knowledgesheet (Gupta & Akhter, 2000) que é uma folha de cálculo dedutiva que apresenta um interface fácil de usar, permitindo restrições de domínio finito (por exemplo, o conjunto finito X é o domínio de uma função f, e Y é o conjunto de valores de f, onde o número de elementos de Y não pode exceder o número de elementos de X) e com a finalidade de resolver problemas da satisfação de restrições, cuja especificação e solução está em conformidade com uma estrutura de 2 dimensões, por exemplo, problemas de programação, horários, etc.

No Knowledgesheet, o utilizador atribui restrições sobre as células e associa cada uma a um domínio finito, podendo também atribuir-lhes valores específicos. Uma vez satisfeito com as restrições, o utilizador dá a instrução para que as restrições sejam compiladas num programa de lógica de restrições e resolvidas. Há duas diferenças entre este sistema e os anteriores. Primeiro, a propagação não ocorre automaticamente pois os valores são assignados a células. Segundo, as restrições são criadas e posteriormente removidas. Esta situação é ideal para grandes problemas que demoram alguns minutos ou até mais tempo a resolver.

Em 2003 surge uma folha de cálculo dedutiva com capacidades similares ao Knowledgesheet: o CSSOLVER (Felfernig et al., 2003). Mas este sistema, em vez de usar um mecanismo de programação lógica para resolver as restrições, usa um mecanismo de programação de restrições. Para encontrar uma solução que simplesmente satisfaça as restrições, o CSSOLVER permite ao utilizador especificar uma função de otimização e encontrar uma solução ótima. O objetivo principal de Felfernig passava sobretudo por criar uma ferramenta que permitisse esconder os detalhes técnicos do processo de raciocínio subjacente aos problemas a resolver, com a ajuda da maior simplicidade de utilização da folha de cálculo.

Em linhas gerais, a forma de usar o CSSOLVER começa pelo utilizador definir as variáveis do problema e o seu domínio. Este pode ter um intervalo inteiro ou uma enumeração de valores simbólicos numa área especificada. O programador pode especificar os *inputs* para as variáveis do problema e o utilizador final pode selecionar esses inputs, referenciando para outras partes da folha de cálculo. Seguidamente o programador define as restrições do problema, podendo fazê-lo de duas formas. Na primeira, os tuplos de valores compatíveis para variáveis individuais são naturalmente dados em tabelas de compatibilidade, ou seja, a restrição é definida de forma explícita enumerando todas as constelações permitidas. Na segunda opção, o utilizador pode

indicar as restrições usando uma linguagem de restrições que tem a mesma sintaxe que a linguagem padrão no ambiente da folha de cálculo (Felfernig et al., 2003).

Por exemplo:

A seguinte expressão funcional condicional *if < cond > then <val1> else <val2>* é expressa no Excel: *IF(cond;val1;val2).*

Toda a informação relativa às restrições do problema é colocada em locais arbitrários da folha.

O utilizador pode de forma arbitrária criar o *layout* do aplicativo na área de gestão onde são definidas as regiões que contêm o problema de satisfação de restrições. O ambiente padrão da folha é alargado com um menu, que permite ao programador definir a localização da área de gestão e iniciar a pesquisa ou o processo de otimização. Se o CSSOLVER encontra uma atribuição consistente, o resultado é exibido na área de saída.

Posteriormente é desenvolvido o PrediCalc (Kassoff et al., 2005), uma folha de cálculo dedutiva concebida para resolução de problemas de satisfação de restrições interativa. O problema de satisfação de restrições (CSP - *Constraint Satisfaction Problems*) diz respeito a um conjunto de objetos cujos estados devem satisfazer uma série de restrições. A interface do PrediCalc é diferente da das folhas de cálculo tradicionais. Tem uma tela em branco, um editor de restrições textual e um editor de domínio.

No PrediCalc, o estado é definido por exemplo por variáveis *Xi* com valores do domínio *Di*. Sobre eles são efetuados testes objetivos que são conjuntos de restrições que especificam combinações possíveis de valores para subconjuntos de variáveis. As restrições no PrediCalc são escritas em lógica de primeira ordem (LPO), conhecida também como cálculo de predicados de primeira ordem (CPPO) que é um sistema lógico que estende a lógica proposicional. O PrediCalc permite restrições lógicas gerais e propagação multi-direcional.

Exemplo:

No caso de A1 = A2 + A3

Ao serem especificados valores para A2 e A3, a folha calcula A1. No entanto numa folha de cálculo tradicional não é possível atribuir um valor a A1 e A2 por exemplo e obter A3.

A PrediCalc, como definida pelos seus criadores, tem aplicação em gestão de dados, design e configuração. Michael Kassoff dá o exemplo de uma sistema de gestão de salas usando a PrediCalc.

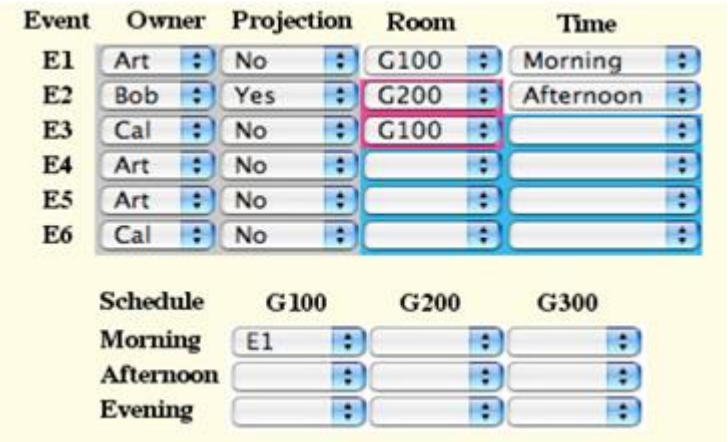


Figura 10 - Parte de um sistema de gestão de salas criado através da PrediCalc (Fonte: Kassoff et al., 2005).

O PrediCalc permite inconsistência (argumentos logicamente inconsistentes, sem fundamento, inválido ou falha na tentativa de provar o que está a ser alegado) entre as atribuições de valor e as restrições. Esta abordagem difere das técnicas de manutenção de consistência tradicionais. Além disso, o PrediCalc mostra as consequências das atribuições de valores, mesmo quando as atribuições são inconsistentes com as restrições. Outro aspeto importante do PrediCalc é a sua organização de base de dados de células da folha em tabelas. Como o PrediCalc usa nomes estruturados para as células, estas tabelas podem ser consultadas como se fossem tabelas de base de dados, mantendo a capacidade de se referir a uma célula individualmente pelo seu nome. Michael Kassoff refere que o PrediCalc preenche a lacuna entre as bases de dados tradicionais e as folhas de cálculo e ainda relativamente à situação da multi-direcionalidade e das restrições de muitos para muitos (nas restrições estruturais temos que o rácio de cardinalidade de restrições define o número de instâncias de relações que uma entidade pode participar na relação com outras entidades, podendo ser de um para um, um para muitos, muitos para um e muitos para muitos**)**. Existem outros sistemas que o permitem, no entanto, nenhum desses sistemas permite a propagação sob inconsistência ou usando nomes estruturados para células.

Em 2004 a DARPA (*Defense Advanced Research Projects Agency*) decidiu organizar um evento em Arlington, Virgínia (EUA), ao qual chamou "*Small Business Innovation Research*", que consistiu na apresentação de propostas sobre folhas de cálculo dedutivas (Gunning, 2004).
A partir daí são fundados 4 projetos: o NEXCEL (Cervesato, 2007), o LESS (Valente et al., 2007) , o XcelLog (Ramakrishnan et al., 2007) e um sistema sem designação de Waltzman (Tallis et al., 2007). Todos estes sistemas vieram expandir as capacidades do Microsoft Excel. A familiaridade que a maioria dos utilizadores finais tem com a folha de cálculo do Excel foi um importante critério para a escolha da DARPA, que pretendia usar as ferramentas para fins militares, tendo folhas de cálculo dedutivas que permitissem no final decisões rápidas e em situações temporalmente críticas.

Iliano Cervesato desenvolveu o NEXCEL (Cervesato, 2007) onde um dos objetivos era criar uma ferramenta para a manipulação de relações como objetos de primeira classe, permitindo aos utilizadores especificarem regiões da folha como tabelas relacionais que podem conter tuplos enários. Como linguagem para as fórmulas relacionais foi criada uma variante da linguagem de programação Datalog que é uma linguagem de consulta baseada na linguagem de programação Prolog (Huang et al., 2011). Os motivos para se basearem no Datalog têm a ver com o facto de este apresentar excelentes propriedades computacionais e expressividade, e vir com uma série de algoritmos tradicionais que são compatíveis com a sua previsível utilização num contexto de folha de cálculo. Estendeu a sua sintaxe de forma a permitir a incorporação de matrizes e fórmulas escalares da folha de cálculo tradicional e também para a tornar utilizável numa ampla gama de outras circunstâncias práticas.
O protótipo do NEXCEL consiste de dois componentes, um módulo *add-in* para Excel 2003 e um motor de inferência dedutivo. O módulo *add-in* foi desenvolvido em Visual Basic for Aplications e atua como ponte entre o Excel e o motor de inferência, reconhecendo as interações do utilizador destinadas a aceder às funcionalidades alargadas e expedindo-as sempre que necessário para o motor de inferência. Ao carregar uma folha de cálculo dedutiva a partir de um arquivo, ele vai comunicar o conteúdo de todas as regiões utilizadas relacionalmente, bem como quaisquer cláusulas definidas que encontrar, e instala os registos avaliados nas células apropriadas assim que as recebe do motor de inferência. Este processo é realizado ao longo de várias iterações. Da mesma forma, vai transmitir qualquer alteração ao fragmento lógico da folha

de cálculo e atualiza os valores deduzidos como reportado pelo mecanismo dedutivo. O sistema vai lidar diretamente apenas com os pedidos mais básicos, como algumas mudanças na geometria de uma relação, por exemplo. Por fim, vai passar sobre os pedidos de explicação e exibir seu resultado. O motor de inferência implementa as funcionalidades de avaliação, atualização e explicação, e mantém estruturas de dados adequadas para estes fins. Incluem uma cópia avaliada de todas as relações referenciadas ou definidas na folha de cálculo dedutiva. Também incluem grafos, rastreando vários tipos de dependências entre predicados, em particular o seu grafo de chamada, que inclui informações de estratificação necessária durante a avaliação.
O protótipo mais recente foi escrito na linguagem funcional Haskell, que combina eficiência e prototipagem rápida.

Em 2007, Andre Valente, David Van Brackle, Hans Chalupský, e Gary Edwards apresentaram o sistema LESS (Valente et al., 2007). O sistema permite que os utilizadores especifiquem as regiões da folha como tabelas relacionais que podem conter tuplos enários. Além disso é bastante flexível nos seus mecanismos de entrada de conhecimento, permitindo às regiões em forma arbitrária da folha serem traduzidas para tuplos enários ou definições de objetos. O LESS é integrado pelo PowerLoom (MacGregor et al., 1997), que segundo MacGregor, é um poderoso sistema lógico de representação de conhecimento e de raciocínio; e pelo Excel. Foi implementado como extensão (*add-in*) para Excel em C++ e Visual Basic. Segundo os autores, o sistema permite tipos expressivos de conhecimento, como declarações meta-nível (ferramentas de produção), termos funcionais, e as regras com submetas complexas e consequentes, além de suportar dedução natural (Valente et al., 2007). Uma das grandes vantagens do LESS é que este adiciona a referência relacional na folha de cálculo além ao invés da tradicional referência posicional. Outras características do LESS são, além das relações explícitas, as relações multi-dimensionais, regras de fácil perceção. No modelo do LESS as funções lógicas são introduzidas como fórmulas.
Na figura 11 pode-se ver a funcionalidade do PowerLoom, disponível nas funções da folha de cálculo, como chamadas de funções.

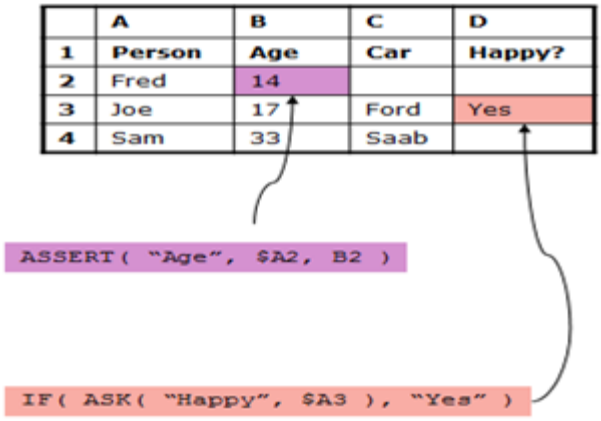


Figura 11 - LESS (Valente et al., 2007).

Outra característica é o caso das tabelas instanciadas, onde um conjunto de instâncias de uma dada classe é representado como uma tabela específica da folha de cálculo. As regras são introduzidas como texto, tal como funções da folha de cálculo. Como referido anteriormente, uma das grandes vantagens do LESS é a referência relacional que usa ao invés da referência posicional das folhas de cálculo. A lógica usa a referência relacional para se referir a informação e o seu significado é transmitido por relações e afirmações (asserções). A informação é referenciada por variáveis em aberto.

Exemplo:
Função lógica (horas? pessoa? mês? Tarefa?)
Significando o número de horas que uma pessoa gasta numa determinada tarefa num mês.

No posicionamento referencial a função SUM(B2:B4) significa a soma de 3 células imediatamente acima, como no exemplo seguinte:

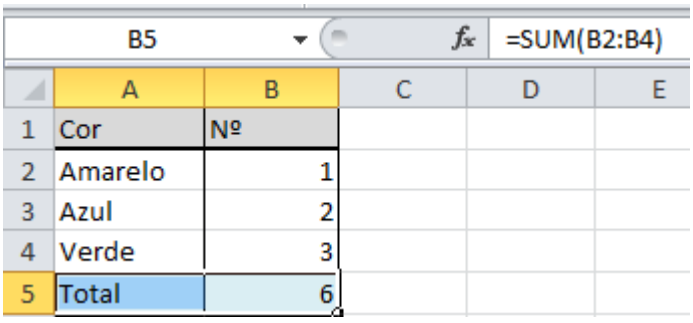


Figura 12 - Exemplo em Excel.

No entanto se a intenção for somar sempre todas as linhas acima, sempre que seja adicionada uma nova linha a soma vai ficar errada, como demonstra o exemplo a seguir:

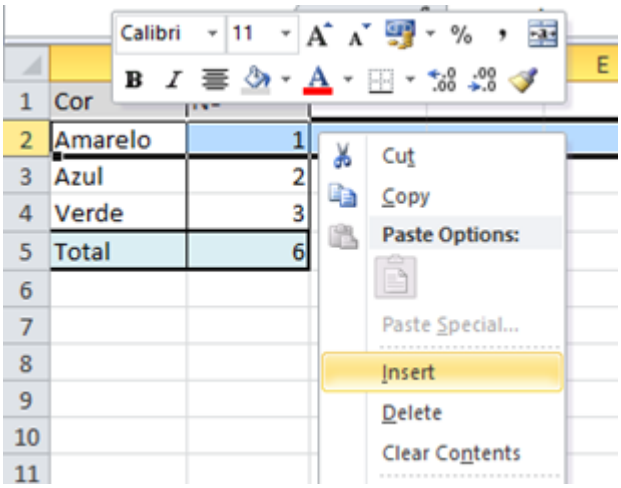


Figura 13 - Exemplo em Excel.

Foi adicionada um novo tuplo (linha) na tabela e o total não atualizou corretamente, como demonstra a figura 14.

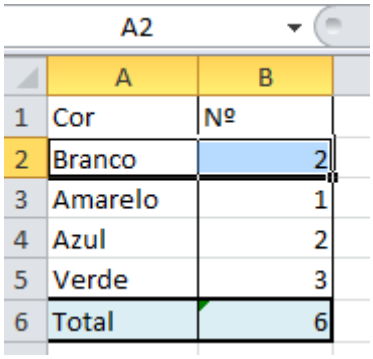


Figura 14 - Exemplo em Excel.

Com o LESS todas as referências são relacionais e portanto por lógica qualquer linha adicionada será contabilizada para o total.

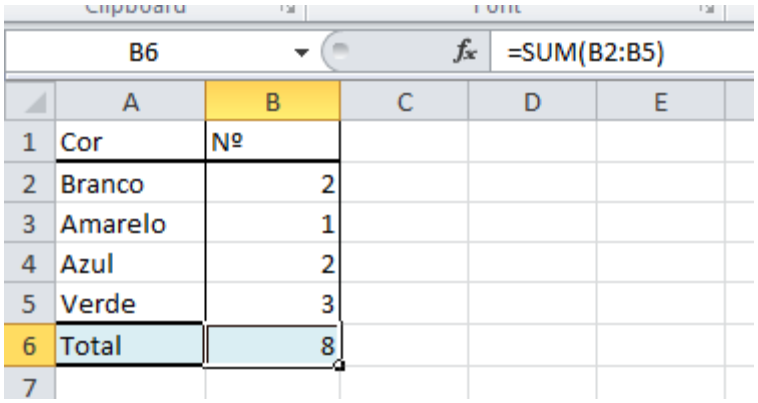


Figura 15 - Exemplo em Excel.

C.R. Ramakrishnan, I.V. Ramakrishnan, e David Warren criaram o XcelLog (Ramaskrishnan et al., 2007), que usam tabelas de classes para representar triplos sujeito-predicado-objeto Foi construído sobre um motor dedutivo de Prolog, embora este fato não seja do conhecimento do utilizador, que escreve fórmulas que utilizam apenas as referências da célula. O resultado é uma interface que permite que o utilizador produza consultas de Datalog combinadas com fórmulas de folhas de cálculo tradicionais.

O Excel serviu de *frontend* enquanto as expressões DSS (*deductive spreadsheet*) foram avaliadas por XSB no *backend* (XSB é um sistema de programação lógica e base de dados dedutivo para Unix e Windows (Warren et al., 1993).

Desta forma, os utilizadores do XcelLog continuam a ter os benefícios da tradicional folha de cálculo, juntamente com o poder adicional de dedução. As células do Excel são usadas no seu estilo tradicional. As células com expressões dentro de "*[ ]*" são tratadas como células DSS, e as expressões são avaliadas pelo sistema XSB.

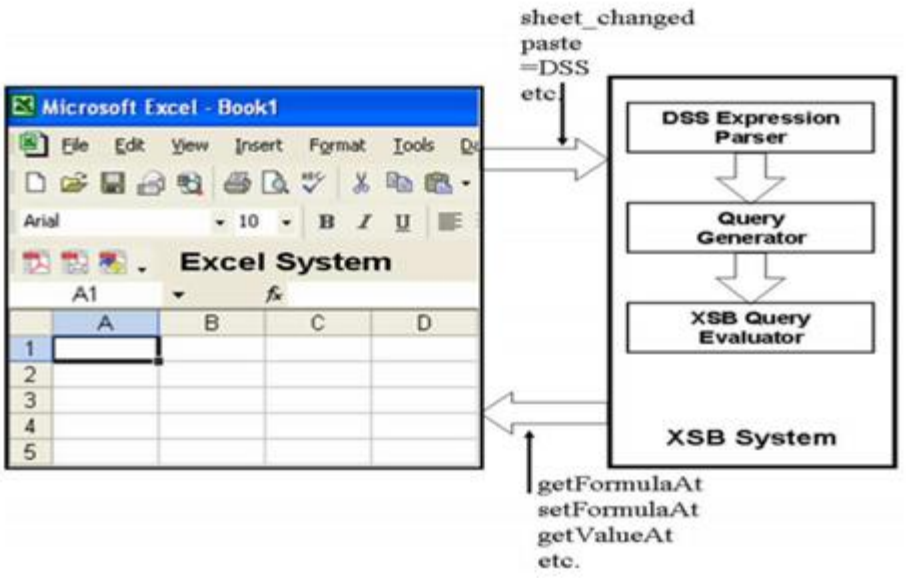


Figura 16 - Sistema DSS do XcelLog (Ramaskrishnan, 2007).

	member	preferred	student	univ
PUB	*PUB.preferred* && *PUB.student* /* Rule 1 */ {Amy}	*ORG.preferred* /* Rule 2 */ {Amy, Joe}	*PUB.univ.student* /* Rule 4 */ {Amy, Bob}	*UAB.member* /* Rule 5 */ {ESU, USB}
ORG		*IEEE.member* /* Rule 3 */ {Amy, Joe}		
IEEE	{Amy, Joe}			
UAB	{ESU, USB}			
ESU			{Amy}	
USB			{Bob}	

Figura 17 - Exemplo da XcelLog (Ramaskrishnan, 2007).

No sistema XcelLog cada célula pode conter um conjunto de valores. As referências das células correspondem a expressões que resultam em conjuntos de valores. Na figura 17 as expressões chamadas de intenções são mostradas em itálico e os seus valores chamados de extensões são mostrados em tele-tipo { } . As expressões e comentários de células (fechados entre "/ *" e "* /") são mostrados para ilustração apenas. Tal como nas folhas de cálculo tradicionais, o utilizador

apenas especifica as intenções e o sistema DSS computa as extensões, mostrando-as nas células e recalcula-os sempre que os valores das células mudam. Considerando a codificação da regra 3 do exemplo da figura 17, que indica que todo o membro *IEEE* é um cliente preferencial da ORG e é específico em DSS utilizando uma célula de referência: a célula *ORG.preferred* contém uma referência para outra célula *IEEE.member*, indicando que o que ocorre em *IEEE.member* também deve ocorrer no *ORG.preferred.* Esta situação é análoga à referência de uma célula para o valor numérico noutra célula numa folha de cálculo tradicional. A regra 4 estabelece que PUB delega a identificação dos estudantes a universidades reconhecidas. *PUB.univ* contém o conjunto de todas as universidades reconhecidas pelo *PUB* e portanto $u.student \subseteq PUB.student$ sempre que $u \in PUB.univ$. A regra 1 refere que *Pub.member* consiste em entidades que estão em ambos PUB.preferred e PUB.student.

O poder da DSS passa assim por permitir que as células contenham múltiplos valores e que permitam referências de células e que declarem que uma célula deve conter todos os elementos de outra célula. A DSS permite ainda que uma célula *a.b* contenha múltiplas referências de células, e o significado de tal expressão é o de que o valor de *a.b* é um conjunto que contém a união de todos os valores das células referidas. Além disso, as referências de célula podem ser recursivas.

Marcelo Tallis, Rand Waltzman, e Bob Balzer (Tallis et al., 2007) apresentaram uma folha de cálculo dedutiva que permite que os utilizadores especifiquem as regiões da folha como tabelas de classes, que permitem a representação natural de triplos sujeito-predicado-objeto. Para demonstrar a sua flexibilidade foi ligado com sucesso a mecanismos de regras Prolog e um mecanismo de regras baseadas em RETE, um algoritmo padrão de correspondência eficiente para a implementação de sistemas de regras de produção (Forgy, 1974).

Os autores referem que desenharam e implementaram uma extensão para Excel que integra este com um motor dedutivo baseado na linguagem ontológica OWL (*Web Ontology Language*) + SWRL (*Semantic Web Rule Language*) (Tallis et al., 2007). A implementação do protótipo inclui uma clara separação entre o *frontend* do mapeamento da lógica dedutiva e o *backend* do servidor lógico que permite alterar o mecanismo dedutivo com o mínimo esforço.

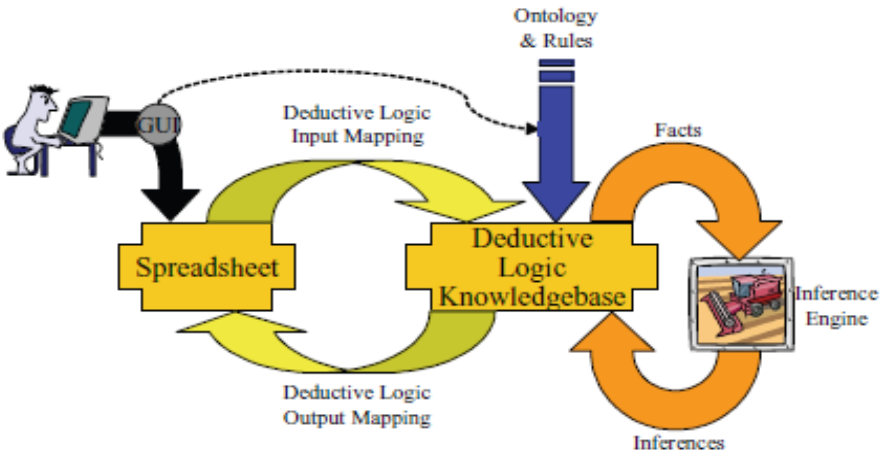


Figura 18 - Arquitetura da folha de cálculo dedutiva (Tallis et al., 2007).

A figura 18 mostra uma tabela da folha de cálculo que lista um grupo de pessoas em que cada linha descreve uma pessoa e cada coluna indica os valores para uma das propriedades da pessoa (por exemplo, género). Alguns valores da tabela podem ser deduzidos através de outros valores na mesma tabela. Por exemplo, os irmãos e irmãs de uma pessoa podem ser deduzidos a partir do gênero, das propriedades do pai e da mãe da pessoa e de outras pessoas na tabela. O objetivo dos autores foi de enriquecer a folha de cálculo com capacidades dedutivas de forma a automaticamente serem carregados os valores na folha, tal como as propriedades irmão e irmã. A ideia passou por fornecer esta capacidade ao sistema enquanto que, ao mesmo tempo, é preservada a forma de interagir com a folha de cálculo por parte dos utilizadores (Tallis et al., 2007).

Assim, introduzindo dados em algumas células da folha (por exemplo, a coluna de *Gender*) verifica-se que se propagam os dados a outras células da folha (por exemplo, a coluna *Brother*).

FirstName	LastName	Gender	Age	Father	Mother	Brother	Sister
John	Smith	Male	42				
Mary	Smith	Female	40				
Bill	Smith	Male	17	John Smith	Mary Smith		Jane Smith
Jane	Smith	Female	15	John Smith	Mary Smith	Bill Smith	
Fred	Jones	Male	82				
Joan	Jones	Female	79				
Bob	Jones	Male	45	Fred Jones	Joan Jones		Sue Jones
Sue	Jones	Female	41	Fred Jones	Joan Jones	Bob Jones	
Sam	Jones	Male	17	Bob Jones	Mary Smith	Marty Jones	Nancy Jones
Marty	Jones	Male	14	Bob Jones	Mary Smith	Sam Jones	Nancy Jones
Nancy	Jones	Female	12	Bob Jones	Mary Smith	Sam Jones Marty Jones	
			Counts =	7	7	6	4

Figura 19 - Folha de cálculo (Tallis et al., 2007).

2.4.2.Matriz de comparação das folhas de cálculo dedutivas

Na validação feita às várias folhas de cálculo dedutivas, não se verificou, uma análise comparativa muito intensiva com base nas várias características variadas de cada uma das folhas analisadas.

Tendo por base que esta comparação é uma importante ferramenta para análise presente e futura do estado atual das folhas de cálculo dedutivas, foi tomada a iniciativa de criar um modelo de comparação que se baseia numa matriz onde são definidas sete características importantes e das mais referidas nos artigos estudados, onde se comparam as ferramentas analisadas.

Assim as características escolhidas para a criação da matriz são: o tipo de ligação existente da folha com o motor dedutivo, a forma como são carregados os tuplos da folha, o interface de cada folha, tipo de restrições permitidas e/ou tipo de referência, sintaxe utilizada, a forma como são especificadas as regiões da folha e uma ultima dimensão onde se agregam características variadas, mas mais específicas de cada folha.

1.O tipo de ligação com o motor dedutivo vem demonstrar como nas diferentes folhas foi feita a comunicação, ou a ponte, entre a simples folha de cálculo tradicional e o motor lógico.

2.Outra dimensão escolhida foi a forma como são carregados os tuplos (linhas) da folha, pois esta característica varia também de folha para folha.

3.Uma dimensão bastante importante é o interface das folhas, que sofreu evolução desde os seus primórdios, mas sobretudo, para se perceber se essa evolução vai facilitar a usabilidade dos utilizadores.

4.Tipo de restrições permitidas e/ou tipo de referência que é uma dimensão que visa perceber a evolução em termos das regras e restrições nas folhas de cálculo dedutivas, assim como também se verifica evolução ao longo dos tempos na referência, que pode essencialmente ser posicional ou relacional .

5.Outra característica que é analisada é a sintaxe a usar na folha, de forma a se perceber o grau de complexidade de interação do utilizador ao dar instruções ou fazer pedidos (questões) na folha.

6.Outra importante característica de valor é a forma como são especificadas as regiões da folha que definem o tratamento que cada uma dá à informação que é guardada como base de conhecimento.
Esta informação pode variar e a forma como se apresenta nas diferentes folhas vem mostrar como se podem selecionar ou referir as áreas da folha com tabelas de base de dados e regras (base de conhecimento), podendo nalguns casos ser mais restrita e noutros apresentar possibilidade de especificação até outras folhas.

7.A sétima dimensão visa analisar outras características que não são de fácil comparação, mas essencialmente que possam ser especificidades de determinadas folhas.

Para as folha onde não existe a dimensão em questão, ou não é referenciada nos artigos lidos foi colocado o símbolo “-”.

Tabela 1 - Matriz de comparação das folhas de cálculo dedutivas

Sistema	1.Ligação com motor dedutivo	2.Carregamento de tuplos	3.Interface	4.Tipo de restrições permitidas e/ou tipo de referenciação	5.Sintaxe usada pelo utilizador	6.Especificação regiões da folha	7.Outras Características
LogiCalc (Kriwaczek, 1988)	Regras traduzidas internamente para Prolog	Os tuplos (linhas) são carregados numa única célula	-	Apenas conjuntivas	Sintaxe da folha de cálculo tradicional	Guardadas como relações da base de dados	Consultas que apenas podem apresentar conjunções
van Emden et al. 1986	-	-	O interface consiste numa instalação de consultas incrementais acopladas numa matriz.	Apenas conjuntivas	Sintaxe da folha de cálculo tradicional	Guardadas como relações da base de dados	Queries incrementais
PERPLEX (Spenke & Beiken 1986)	-	-	-	Restrições com base em predicados internos	Sintaxe da folha de cálculo tradicional	Permite a criação de regras usando as relações das bases de dados.	-
Knowledgesheet (Gupta & Akhter, 2000)	Ligação a um motor de programação lógica de restrições de domínio finito	-	O interface tem 2 partes principais: área de dados e janela de visualização com scrolling.	Restriçoes de domínio finito	Sintaxe da folha de cálculo tradicional	-	Usa um 'constraint programming engine'. Propagação não ocorre automaticamente.
CSSOLVER (Felfernig et al., 2003)	A comunicação entre a folha de cálculo e o "Core Solver" é baseado num interface COM (Component Object Model)	-	O posicionamento e layout dos campos de entrada do utilizador podem ser escolhidos arbitrariamente pelo programador	Restriçoes de domínio finito e permite referenciação para outras partes da folha de cálculo	Linguagem padrão no ambiente da folha de cálculo	Toda a informação do problema de restrição é colocada em locais arbitrários da folha	Usa um 'constraint programming engine'
PrediCalc (Kassoff et al., 2005)	-	-	Tem uma tela em branco, um editor de restrições textual e um editor de domínio.	Permite restrições lógicas gerais; permite inconsistência entre as atribuições de valor e as restrições	Semelhante aos nomes de predicados estruturados permitidos na linguagem de programação lógica HiLog. As fórmulas podem ser construídas a partir destas estrutura de nomes e os conectivos lógicos habituais "e", "ou", "não", "=>", "<=", e "<=>" e os quantificadores for all (para todo) e exists (existe um).	Organização de base de dados de células em tabelas	Restrições de muitos para muitos. Propagação multi-direcional; propagação sob inconsistência.

Sistema	1.Ligação com motor dedutivo	2.Carregamento de tuplos	3.Interface	4.Tipo de restrições permitidas e/ou tipo de referenciação	5.Sintaxe usada pelo utilizador	6.Especificação regiões da folha	7.Outras características
NEXCEL (Cervesato, 2007)	Módulo *add-in* foi desenvolvido em Visual Basic for Aplications e actua como ponte entre o Excel e o motor de inferência	Tem uma componente que explica ao utilizador porque um certo tuplo foi ou não calculado.	Tem um assistente de inserção de fórmulas para suportar as cláusulas que definem. O mecanismo tradicional "copy-paste" das folhas de cálculo para propagação.	-	É usada uma variante da linguagem de programação Datalog	-	Tem uma componente que explica ao utilizador porque um certo tuplo foi ou não calculado.
LESS (Valente et al., 2007)	Extensão *add-in* para Excel em C++ e Visual Basic	-	Tabelas de instâncias de uma determinada classe são representadas como tabelas especiais da folha e são criadas novas tabelas de instâncias de forma equivalente à criação de uma classe nas regras numa ontologia . As regras e funções são inseridos como texto.	Referência relacional	As funções lógicas são introduzidas como fórmulas	Especifica as regiões da folha como tabelas relacionais que podem conter tuplas n-árias	Permite tipos expressivos de conhecimento, como declarações meta-nível (ferramentas de produção), termos funcionais, e as regras com submetas complexas e consequentes, e suporta dedução natural, e raciocínio limitado de ordem superior. Permite relações multidimensionais
XcelLog (Ramaskrishnan et al., 2007)	O Excel serviu de frontend enquanto as expressões DSS (deductive spreadsheet) foram avaliadas por XSB no backend	Usa valores como tuplos nas células	As células do Excel são usadas no seu estilo tradicional. As células com expressões dentro de "[ ]" são tratadas como células DSS	Permitir que as células contenham multiplos valores e que permitam referências de células declarem que uma célula deve conter todos os elementos de outra célula.	Consultas de Datalog misturados com fórmulas de folhas de cálculo tradicionais.	Especifica as regiões da folha como tabelas relacionais que podem conter tuplas n-árias	Permite células definidas recusivamente
(Tallis et al., 2007)	Extensão para Excel que integra este com um motor dedutivo baseado na linguagem ontológica OWL + SWRL.	-	As células do Excel são usadas no seu estilo tradicional.	-	Sintaxe da folha de cálculo tradicional	As regiões da folha como tabelas de classes, que permitem a representação natural de triplos sujeito-predicado-objeto.	Foi ligado com sucesso a mecanismos de regras Prolog e um mecanismo de regras baseadas em Rete

2.4.3.Considerações sobre a evolução das folhas de cálculo dedutivas

Em jeito de conclusão quanto à evolução das folhas de cálculo dedutivas, são uma área de pesquisa e desenvolvimento bastante interessantes e úteis para vários ramos de atividade (Cervesato, 2007).

A implementação de folhas de cálculo dedutivas difere tendo em conta o tipo de desenvolvimento pretendido e a quem se pretende beneficiar. Os sistemas analisados têm a intenção de estender ou melhor as folhas de cálculo, mas outros têm um propósito de melhorar os sistemas lógicos. Enquanto alguns avanços tentam separar-se mais do paradigma da folha de cálculo, outros tentam manter o máximo de características tradicionais acrescentado-lhe características de sistemas lógicos, sem nunca deixar cair, no entanto, as capacidades que tornaram as folhas de cálculo tradicionais tão populares.

O trabalho a realizar reside essencialmente em tornar tarefas complexas em tarefas simples para os utilizadores finais, permitindo que sejam fáceis de manejar pela maioria dos utilizadores.

Na pesquisa intensiva efetuada verificou-se que nos últimos anos não houve grandes avanços nem novos sistemas (que se tenha conseguido encontrar nas pesquisas efetuadas) relativamente aos últimos desenvolvimentos que se verificaram por volta de 2007. Foi escrito um livro em 2013 por Iliano Cervesato, mas que serve em jeito de aglomerado do que foi feito, resumo e esclarecimento detalhado técnico sobre folhas de cálculo dedutivas.

Esta situação deixa algumas questões no ar, tendo em conta o potencial da temática em questão, verificando-se uma estagnação no desenvolvimento deste tipo de sistemas e mesmo na sua investigação. A razão para tal continua para nós uma incógnita e não se especulam justificações para tal acontecimento. Todo o trabalho, mas principalmente o próximo capítulo apresentado nesta dissertação, tentam trazer alguma dinâmica de volta a este tema.

Capítulo III - A ferramenta Xprolog

Este capítulo inicia-se com uma descrição mais genérica da ferramenta desenvolvida, seguindo-se uma descrição mais técnica e detalhada sobre o seu desenvolvimento e funcionamento. Seguem-se apresentações e exemplos das principais características da ferramenta, recorrendo a uma espécie de tutorial de iniciação. Por fim, as vantagens e limitações da sua utilização são exemplificadas.

3.1.Descrição da ferramenta

Para a ferramenta criada, foi escolhido a designação de Xprolog (lê-se “xis Prolog”). A opção foi feita por esta ser simples de pronunciar e escrever, assim como considerada de fácil memorização por parte dos utilizadores.

A ferramenta criada pode ser vista como um suplemento para o Excel. Como já referido anteriormente, a escolha foi feita tendo por base a vasta comunidade dos seus utilizadores, a facilidade e utilidade da sua utilização, mesmo a um nível básico, e por ser possível, em termos técnicos, estabelecer a ligação com sistemas Prolog, questão que será melhor esclarecida mais à frente na abordagem técnica.

O Xprolog tem a função principal de facilitar a um utilizador básico de Excel efetuar questões dedutivas sobre bases de conhecimento e obter instantaneamente as respostas, ou seja, potenciar o uso de raciocínio lógico em plataformas como o Excel, com todas as suas vantagens inerentes já referidas.

Assim sendo, o Xprolog possibilita uma utilização exatamente igual à do Excel tradicional mas permite ainda a resolução de problemas que requeiram outras capacidades, como a dedução, sem alterar a facilidade e o estilo de utilização do Excel original. O utilizador pode especificar bases de conhecimento nas regiões da folha de cálculo que entender, pode invocar as capacidades de dedução do Prolog para obter resultados necessários, e pode usar os resultados obtidos na resolução de outros problemas. As bases de conhecimento referidas são constituídas por factos e

por regras. Cada facto corresponde a um tuplo constituído por uma ou mais células, usando a sintaxe original do Excel. Cada regra é especificada com uma sintaxe baseada na do Prolog, mas recorrendo sempre à interface gráfica usual do Excel.

Tendo em conta que esta situação é uma melhoria ao funcionamento normal de uma folha de cálculo, o Xprolog, apesar de não ser um produto totalmente inovador, vem por um lado alavancar os desenvolvimentos já existentes, apesar de ainda pouco conhecidos e divulgados, assim como tenta tornar mais simples a forma como o utilizador pode ter acesso e resolver, eficazmente e com simplicidade, problemas envolvendo lógica e dedução continuando a usufruir, sem limites, das vantagens inerentes ao Excel, nomeadamente as suas capacidades computacionais tradicionais e todo o seu interface, garantindo elevada usabilidade.

3.2.Principais características da ferramenta

Como se pode verificar na figura 20, no Excel é carregado ou criado o conjunto de factos da base de conhecimento, exatamente como uma tabela tradicional. A 2ª linha com "homem", "mulher" e "filho" corresponde aos nomes dos predicados em Prolog: homem/1, mulher/1 e filho/2.

base de conhecimento			
homem	mulher	filho	
Rui	Maria	Pedro	Rui
Pedro	Joana	Rui	Paulo
		Rui	Maria
		Pedro	Joana
Paulo			

Figura 20 – Factos da base de conhecimento no Excel.

Nas linhas seguintes estão os dados que fazem o papel dos argumentos nos vários factos desses predicados. Por exemplo, a segunda linha da tabela é equivalente aos seguintes factos Prolog: homem('Rui'), mulher('Maria'), filho('Pedro', 'Rui').

Cada predicado pode ter vários argumentos. Além disso, como se pode verificar na figura 20, pode ter-se células em branco entremeadas entre os dados dos vários factos, desde que a região

de factos seja definida corretamente. Assim, a primeira coluna de "homem" corresponde aos seguintes factos Prolog:

homem(rui).
homem(pedro).
homem(paulo).

Os predicados com mais do que um argumento são especificados em células que se estendem sobre várias colunas, como acontece com o predicado filho. "Filho" foi colocada num espaço correspondente a 2 colunas, através do comando "Unir e Centrar" do Excel. As linhas abaixo deste predicado correspondem aos dados dos seus argumentos. A tabela filho na folha Excel é interpretada, em Prolog, como:

filho(pedro, rui).
filho(rui, paulo).
filho(rui, maria).
filho(pedro, joana).

A interpretação neste caso será então: 'Pedro é filho de Rui', Rui é filho de Paulo, etc.

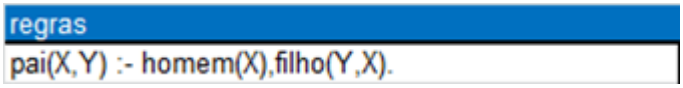


Figura 21 – Regras da base de conhecimento.

Na figura 21 são definidas as regras em Prolog.
Tal como em Prolog, o ":-" significa "se" e o "," significa "e", ou seja, significa uma conjunção, neste caso temos 'X é pai de Y se X é homem e Y é filho de X'.

Figura 22 – Interrogação / Questão colocada no Excel.

Na figura 22 é demonstrada uma interrogação que é colocada pelo utilizador, também em Prolog, sobre a base de conhecimento definida nas figuras 20 e 21. Neste campo é importante ter em consideração que os nomes "rui" e "pedro" devem começar obrigatoriamente por letra minúscula, caso contrário o sistema irá interpretar que se estão a colocar variáveis e não os nomes já existentes na base de conhecimento. Esta limitação é referida mais a frente nas características e limitações do sistema atual.

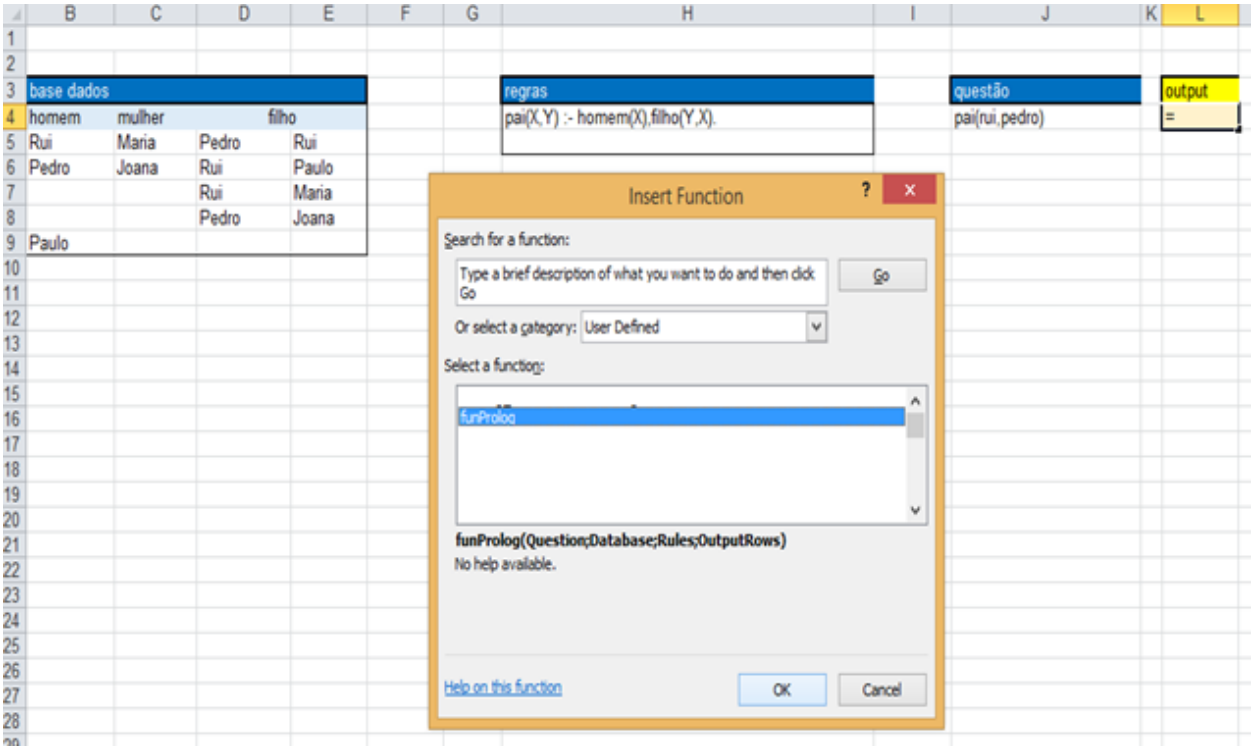


Figura 23 – Inserir a função "*funProlog*".

Na figura 23 verifica-se a inserção da função "*funProlog*" depois de selecionar a área de *Output* (a amarelo na figura 23).

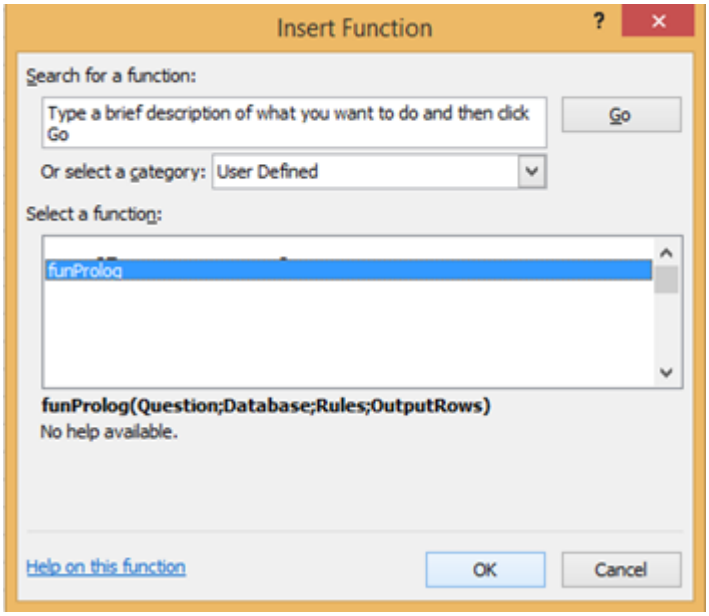


Figura 24 – Pormenor da janela "Insert Function" com a função do Xprolog: "*funProlog*".

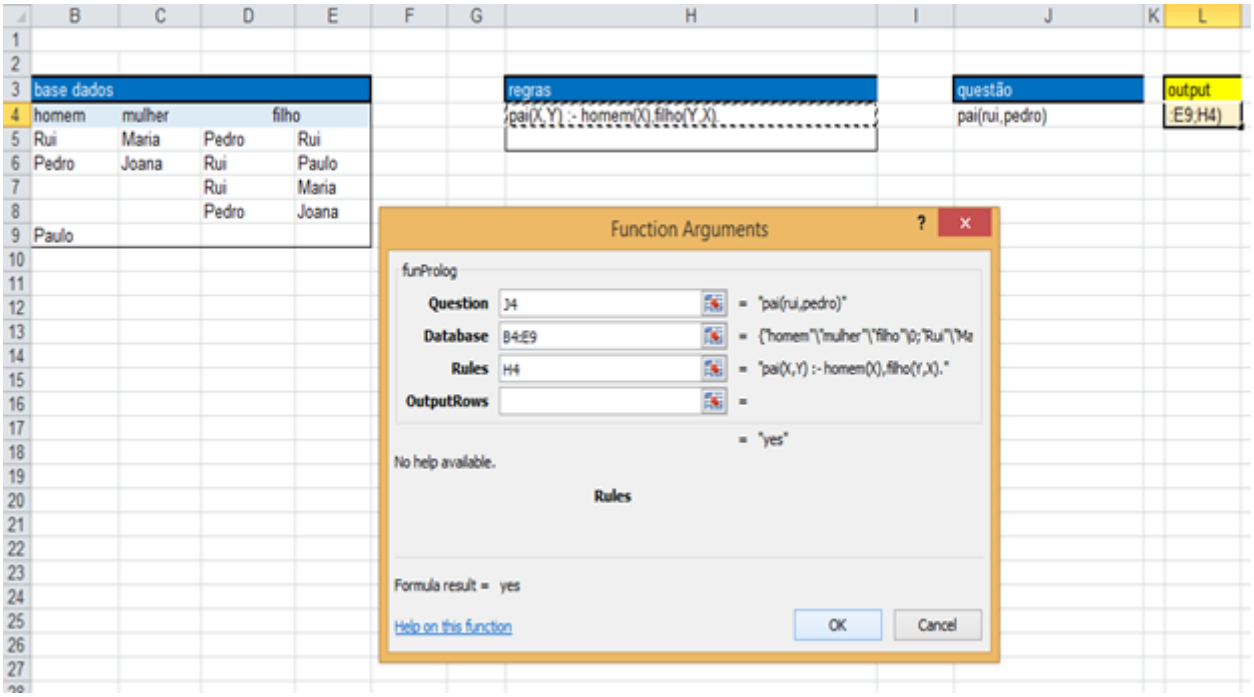


Figura 25 - Preenchimento dos campos da janela auxiliar "Function Arguments" no XProlog.

Na figura 25 pode verificar-se na janela "*Function Arguments*" que os dados carregados no Excel devem agora ser associados aos campos da janela de forma ao sistema efetuar o processamento

da informação e responder à questão colocada. Tal como referem as designações anteriores aos campos a preencher, na "*Question*" é colocada célula no Excel onde está a Questão, na "Database" é colocada a área correspondente à base de conhecimento, "Rules" é colocada a ou as regras definidas. O campo "*OutputRows*" deve ficar sempre por preencher, ou seja, em vazio.
Como se pode verificar na figura 25, a resposta é já pré-visualizada, neste caso "yes" aparece na janela de "Function Arguments", após se carregar os campos com as respetivas células.

Figura 26 - Questão e *Output* no XProlog.

Na figura 26 é visualizada a área de *output* com a resposta à interrogação colocada.

Podemos dividir o tipo de respostas em dois: o primeiro do tipo "*yes/no*", ou seja, é colocada uma questão que se pretende validar ser ou não verdadeira (pergunta fechada). O segundo tipo é de resposta "aberta" onde a área de *output* pode variar de tamanho e forma (pergunta aberta).
Segue um exemplo de utilização com uma resposta que origina uma área de *output* de tamanho não previsível. Neste caso, ao contrário do exemplo anterior onde se sabia que a área de *output* seria sempre uma só célula, podem ser várias, dependendo da questão colocada e dos dados na base de conhecimento.

Base Conhecimento - Clientes Banco				
empresa	particular	valor	deposito	
Alfa	Rui	1000	1000	Rui
Beta	Marco	2000	5000	Marco
	Ana	5000	2000	Ana
			40000	Alfa
		30000	30000	Beta
		40000		

Figura 27 - Base de conhecimento dos clientes do banco.

Na figura 27 está representada uma base de conhecimento em Excel que é interpretada em Prolog da seguinte forma:

empresa(alfa).
empresa(beta).
particular(rui).
particular(marco).
particular(ana).
valor(1000)
valor(2000)
valor(5000)
valor(30000)
valor(40000)
deposito(rui, 1000)
deposito(marco, 5000)
deposito(ana, 2000)
deposito(alfa, 40000)
deposito(beta, 30000)

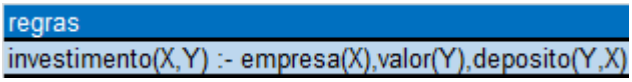


Figura 28 - regras definidas para a "cabeça": investimento.

Na figura 28 pretende-se descrever que X é investimento de Y se X é uma empresa, e Y é uma valor e Y é um depósito de X.

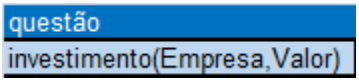


Figura 29 - Interrogação sobre os valores investidos por cada empresa.

Na figura 29, é colocada a questão sobre o valor investido por cada empresa. O sistema deve filtrar inicialmente as empresas e depois o valor investido por cada uma. Como se pode verificar

os argumentos do predicado "investimento" iniciam-se com letra maiúscula de forma a serem interpretados pelo XProlog como variáveis, pois podemos ter várias empresas e vários valores.

questão	output	
investimento(Empresa,Valor)	**Empresa**	**Valor**
	alfa	40000
	beta	30000

Figura 30 - *Output* da interrogação "investimento".

Como se pode verificar na figura 30, apesar de apenas se ter selecionado uma célula para o *output*, o XProlog vai automaticamente preencher tantas células quantas as necessárias para dar a resposta correta à interrogação colocada.

Como se pode constatar nos exemplos anteriores, o que está definido na ferramenta é que a questão deverá ser em linguagem Prolog, pelo que apesar de simples, terá de ser efetuada uma questão com a lógica de programação Prolog. Assim, por exemplo, se temos uma base de conhecimento que diz que Jorge é pai de Mário e de Rui, devemos colocar a questão "pai(jorge, rui)" onde a resposta será afirmativa "Yes". Por fim é escolhida a área na folha de cálculo onde se pretende que seja obtido o *output*, ou seja, a resposta à questão colocada.

3.3.Abordagem Técnica

Para melhor clarificar a abordagem técnica e os processos de criação e de execução do Xprolog, é essencial clarificar alguns conceitos e ferramentas que não são do conhecimento geral.
Uma ferramenta que é utilizada pelo Xprolog e é referida como é feita essa utilização mais à frente, é o SWI-Prolog. A escolha deste *software* deve-se essencialmente a este ser uma implementação em código aberto da linguagem de programação Prolog, largamente compatível com a ISO Prolog, e que é amplamente utilizado em investigação, educação e em aplicações comerciais. Segundo se refere na sua página web, o número de pessoas que utilizam o SWI-Prolog estende-se a mais de um milhão *(http://www.swi-prolog.org/)*. Além disso na sua página

web existem também imensos recursos de ajuda a programadores, assim como uma FAQ que rapidamente ajuda nas dúvidas da comunidade.

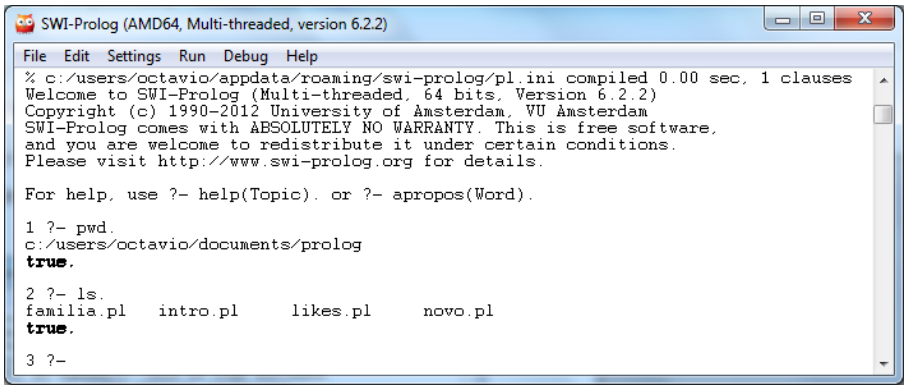


Figura 31 - Software SWI-Prolog (versão 6.2.2).

O sistema foi criado em Visual Basic for Applications (VBA), que é uma implementação da linguagem de programação Visual Basic (VB) da Microsoft e que está incorporada em todos os programas do Microsoft Office, em particular no Excel. Esta implementação pode ser usada para controlar quase todos os aspetos da aplicação, incluindo a manipulação de características de interface do utilizador como menus, barra de ferramentas e criação de formulários.

Assim o VBA adiciona a capacidade de automatizar tarefas no Excel e permitir funcionalidades definidas pelo utilizador. A gravação de macros pode produzir código VBA que replica ações do utilizador, permitindo automatismos simples de tarefas quotidianas. Pode no entanto, como acontece com o XProlog, ser usado o VBA para controlar uma aplicação a partir de outra.

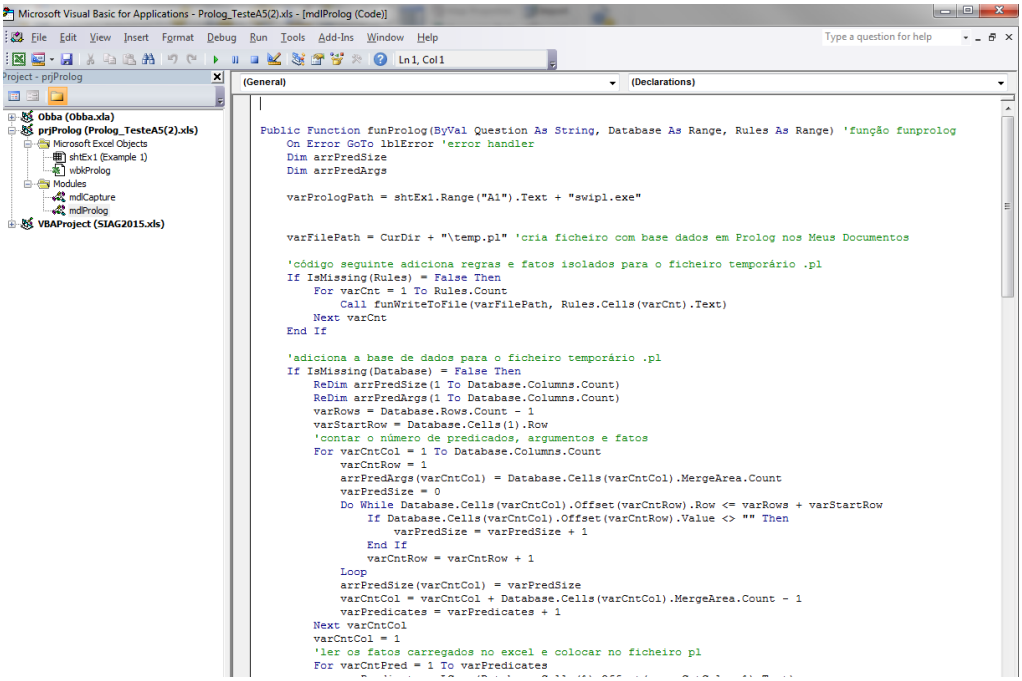


Figura 32 - Exemplo de VBA com código do Xprolog.

Outro conceito que importa clarificar é o de interpretador de comandos, ou seja, programas de computador que são responsáveis por executar ações de acordo com orientações do utilizador via comunicação textual. Assim, uma funcionalidade de um interpretador de comandos é a designada Linha de Comandos, que não passa de uma interface textual que interpreta os comandos do utilizador e os envia ao núcleo (ou *kernel*) da ferramenta para serem executados, imprimindo o resultado posteriormente no ecrã.

Esclarecidos os conceitos das componentes intervenientes no processo do XProlog, podemos agora claramente explicar como a ferramenta funciona.

A implementação e o funcionamento do Xprolog envolve uma ligação entre o VBA do Excel e o SWI-Prolog. Através de um módulo VBA criado por Dipak Audy (2012) (*https://friendpaste.com/3aNELWiC1ed4bzApDhYIPl*) é feita a comunicação entre o Excel e o SWI-Prolog. Depois foi desenvolvido um módulo suplementar que ajuda o Excel a perceber os dados que recebe e a forma como é criado o ficheiro temporário que o SWI-Prolog vai receber.Os factos, as regras e as interrogações inseridos no ficheiro Excel são enviados para esse ficheiro temporário, ao mesmo tempo o SWI-Prolog é executado (em plano de fundo) e recebe esse ficheiro com instruções sobre o que fazer. No ficheiro temporário vão as informações da

base de dados, regras definidas e questões colocadas. O SWI-Prolog lê o ficheiro e responde. Essa resposta é devolvida instantaneamente para o Excel e apresentada na folha do XProlog na área de *output* definida inicialmente pelo utilizador, como vai ser exemplificado mais à frente.

É importante salientar alguns pormenores, tais como o fato do utilizador do sistema não visualizar o ficheiro temporário que é criado e eliminado após a sua utilização pelo sistema, assim como não vai visualizar a execução do SWI-Prolog. Com esta medida pretende-se poupar o utilizador, dispensando a sua intervenção em processos que não dizem respeito aos seus domínio e nível de operação. O utilizador apenas necessita instalar o SWI-Prolog na máquina onde vai correr o XProlog. Este pode ser considerado o único pré-requisito. Todo o processo é assegurado automaticamente pela ferramenta de forma limpa e rápida. Um utilizador típico de Excel, embora necessite de conhecimentos básicos de Prolog para realizar tarefas simples, pode facilmente, apenas abrindo uma simples folha de cálculo, proceder à realização de tarefas e execução de procedimentos de processamento simbólico e inferência que de outra forma apenas era possível, por exemplo programando em Prolog num sistema como o SWI-Prolog, transportando os dados e os resultados entre o Excel e o Prolog.

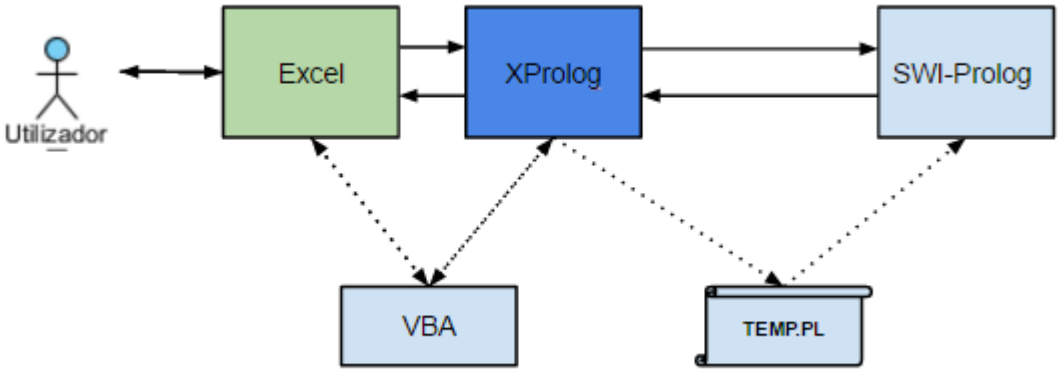


Figura 33 – Arquitetura do processo de funcionamento da ferramenta XProlog.

No diagrama da figura 33 o processo inicia-se com a interação do utilizador com uma página de Excel, carregando ou criando a base de conhecimento (factos e regras) e definindo as interrogações. De seguida o utilizador define a função "*funprolog*" onde vai designar as áreas na folha de Excel correspondentes à base de dados (factos), regras e interrogações, assim como qual a área na folha onde vai ser retornado o *output*. Esta informação é lida pelo XProlog,

implementado em VBA, através do ficheiro temporário *Temp.pl* a representação em Prolog de todas as especificações relevantes contidas na página Excel. Este ficheiro é interpretado pelo SWI-Prolog, que corre em plano de fundo (*background*) no sistema operativo, e que responde às interrogações colocadas. A resposta é devolvida para o Xprolog, que novamente através do VBA a carrega na área de *output* do Excel, definida pelo utilizador.

3.4.Limitações da ferramenta atual

Desenvolver o conceito de folhas de cálculo com capacidades de representação de conhecimento e de raciocínio é o objetivo desta dissertação no entanto, existem ainda algumas limitações na ferramenta desenvolvida que poderão e deverão ser ultrapassadas de futuro de forma a dar garantia à continuidade do projeto. Seguem-se algumas limitações que servem como desafio a novos desenvolvimentos à ferramenta.

A primeira limitação apontada é na escolha da área da base de dados, ou seja, a definição da tabela ou tabelas com os factos. Apesar da área da folha de cálculo poder ser qualquer uma definida pelo utilizador, tem de ser, neste momento, apenas uma única área, ou seja, poderão ser várias tabelas mas terão de ser todas na mesma zona, única, e que é selecionada quando é especificada a função de ligação ao Prolog. Como se pode verificar na figura 34 a linha "3" é a primeira linha da tabela 1 e é a dos predicados e as seguintes os argumentos respetivos. Sendo assim, não é possível definir várias bases de dados em zonas distintas da folha de cálculo e assignar regras e questões sobre as mesmas, ou seja, a tabela 2 não poderia ser usada para a questão que se pretende colocar no exemplo da figura 34 abaixo. Para poder ser uma tabela da base de dados para a questão a colocar teria de estar posicionada ao lado da tabela 1 alinhada pela primeira linha (tuplo 3).

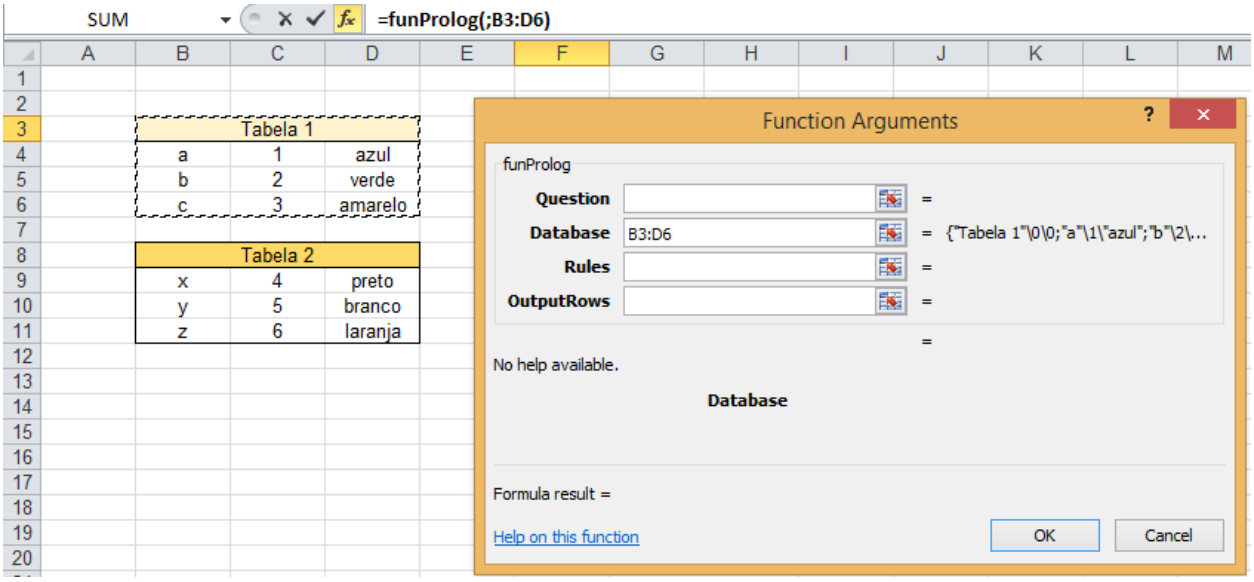


Figura 34 – Exemplo Xprolog.

É no entanto possível numa outra questão a ser colocada selecionar outra zona da folha com outra base de dados, por exemplo onde está a tabela 2 na figura 34 e assim efetuar uma nova função "*funprolog*", mas apenas em relação à nova área selecionada.

Outra limitação é a não utilização de roles, ou seja, na definição das tabelas não é possível designar um role para cada argumento dos predicados a utilizar. No caso da figura 35, verifica-se na tabela 3 um predicado "fatura" que contém três argumentos. No entanto esses argumentos apenas vão ser definidos genericamente nas regras ou nas questões colocadas.

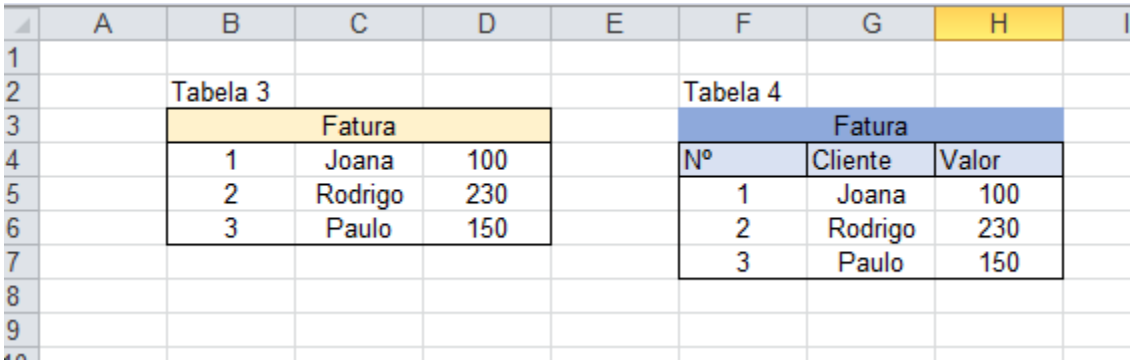


Figura 35 – Exemplo Xprolog.

A tabela 4 na figura 35 não pode neste momento ser interpretada corretamente pelo Xprolog. A definição genérica de cada argumento do predicado "fatura" é uma impossibilidade e portanto uma limitação a ser ultrapassada.

A limitação mais evidente, no entanto é a sintaxe a utilizar. Apesar da simplicidade implícita numa folha de cálculo tradicional em termos de interface e usabilidade, o utilizador necessita sempre de saber programar em Prolog para poder trabalhar com a folha dedutiva apresentada.
A colocação de regras e questões mais simples serão facilmente captadas, mesmo por utilizadores que não programam em Prolog, mas para questões que exijam regras ou interrogações mais complexas, é necessário um conhecimento mais aprofundado da linguagem Prolog.

3.5.Tabela Comparativa das características de folhas de cálculo dedutivas.

Com objetivo de tornar a análise da ferramenta criada mais completa e comparativa com as folhas e cálculo dedutivas analisadas no Capítulo IV foi construída uma tabela com as sete características de análise referidas nesse capítulo e agora analisadas relativamente ao Xprolog.

Tabela 2 - Tabela de características de folhas de cálculo dedutivas relativamente ao Xprolog.

Sistema	**1.Ligação com motor dedutivo**	**2.Carregamento de tuplos**	**3.Interface**	**4.Tipo de restrições permitidas e/ou tipo de referenciação**	**5.Sintaxe usada pelo utilizador**	**6.Especificação regiões da folha**	**7.Outras Características**
Xprolog	Através de uma ligação por linha de comandos entre Excel VBA e o software de lógica SWI-Prolog.	Carregamento de tuplos pode ser por mais do que uma célula	O interface tem 4 partes: (1) base de dados, (2) área de regras, (3) área da interrogação e (4) área de *output* (resposta).	–	Prolog (conceitos básicos). Os resultados pretendidos são obtidos através da fórmula do Xprolog "*funprolog*"	Especificação da base de conhecimento, regras, questões e zona de *output* livres de definição pelo utilizador por toda a folha (s) da "*worksheet*".	Alterações na base de dados, regras ou questões produzem resultados instantâneos na zona de *output* já preenchida.

Capítulo IV - Vantagens da utilização do Xprolog

Neste capítulo são apresentadas as vantagens mais importantes da utilização do Xprolog, assim como a sua comparação com *add-ins* bastante conhecidos e utilizados pelo público em geral.
Uma vantagem importante é a de que o Xprolog não necessita de ser instalado previamente, podendo ser logo usado. Apenas necessita que se tenha o SWI-Prolog instalado na máquina a correr a ferramenta, como já foi referido no capítulo anterior,e ao contrário da maioria dos suplementos encontrados no mercado para Excel que necessitam de ser instalados e que apresentam por vezes pré-requisitos como instalação de pequenos softwares de auxílio a estes suplementos a instalar, como por exemplo o *.NET Framework*, da Microsoft, que é necessário para correr o Power Pivot ou o Power Query que são apresentados seguidamente.

4.1.Suplementos alternativos para Excel

Existem vários suplementos, mais conhecidos como *add-ins* para Excel que permitem estender as capacidades da folha de cálculo tradicional. Estes suplementos permitem adicionar, tal como o Xprolog, capacidades que nas folhas tradicionais não são possíveis, ou mesmo sendo possíveis, são de muito difícil produção. Estes podem fornecer comandos ou funcionalidades para o Excel e que por predefinição não estão imediatamente disponíveis na folha *(https://support.office.com/pt-pt/article/Adicionar-ou-remover-suplementos-0af570c4-5cf3-4fa9-9b88-403625a0b460).*
Dois suplementos bastantes referenciados e que facilmente se podem adquirir são o Power Query e o Power Pivot, que são analisados neste capítulo e exemplificados em comparação com algumas das capacidades do Xprolog. Embora não sejam substitutos completos do Xprolog, pois são ferramentas com capacidades mais úteis para relações entre tabelas, existem situações de comparação que se podem ter em conta ao confrontar problemas que podem ser resolvidos tanto em Xprolog como com a ajuda destes suplementos. No entanto depois de uma análise mais profunda chega-se à conclusão de que o Xprolog pode ser considerado mais um complemento a este tipo de ferramentas e nunca um substituto direto.

4.1.1 – Power Query

O Power Query é uma ferramenta de análise de dados disponibilizada pela Microsoft para integrar no Excel. Com esta ferramenta é possível descobrir, combinar e refinar os dados. Esta ferramenta é particularmente útil como uma aplicação de *Business Intelligence*. É fundamentalmente uma ferramenta que facilita a conexão do Excel com fontes externas de dados. Por exemplo, para fazer uma pesquisa em tabelas de uma base de dados do Microsoft Access, especificar critérios de filtragem e ordenação e retornar o resultado da consulta para uma folha de Excel. Permite combinar dados com origens diversas e divergentes. É possível de seguida fazer cálculos e gráficos com os dados retornados. É portante uma ferramenta que permite essencialmente importar dados de variadas fontes externas tais como páginas web, XML, ficheiros de texto, entre outros (*http://www.microsoft.com/pt-PT/download/details.aspx?id=39379*). Por exemplo, diretamente da folha de Excel pesquisar dados da Wikipédia (tabelas, por exemplo) e importá-los para a folha de Excel, já como tabela de Excel.

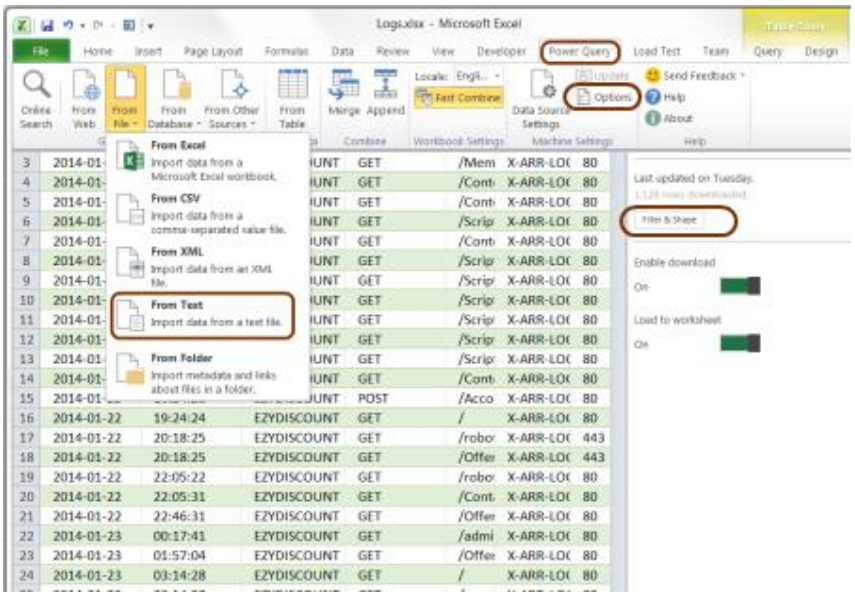


Figura 36 – Power Query.

O Power Query necessita de ser instalado à parte do Excel, o tamanho que ocupa em disco não é muito relevante (cerca de 20 MB). Já as suas implicações em termos de arranque do Excel,

dependendo obviamente da capacidade de memória e processamento da máquina onde está instalado, podem ser significativas.

As desvantagens desta ferramenta face ao Xprolog são a necessidade de um domínio relativamente avançado de Excel para o utilizador poder aproveitar as suas vantagens, a necessidade de ser bem percebido pelo utilizador, requerendo a visualização de tutoriais, e a necessidade de parametrização inicial para poder ser corretamente utilizada.

Outras características que importam salientar e que diferenciam a utilidade desta ferramenta em relação ao XProlog é a de que para se criarem tabelas dinâmicas mais complexas é necessário um outro *add-in* da Microsoft – Power Pivot.

4.1.2. Power Pivot

O Power Pivot é um *add-in* livre, disponibilizado desde 2010 para o Excel e essencialmente estende as capacidades das tabelas dinâmicas no *Excel (http://www.powerpivotpro.com/what-is-powerpivot).*

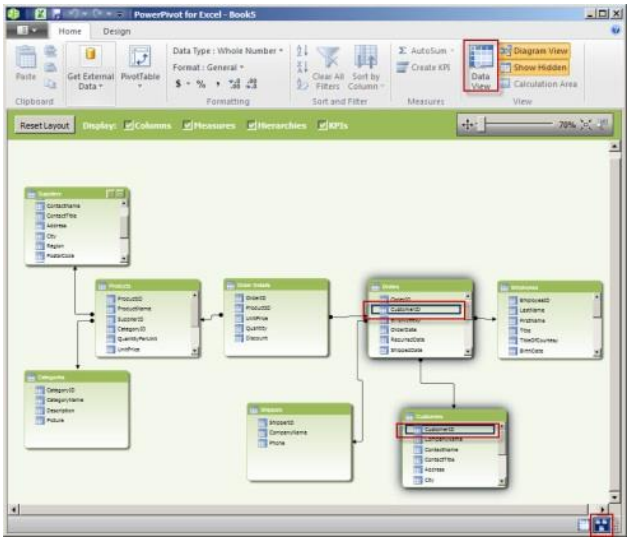


Figura 37 – Power Pivot.

Tem algumas funcionalidades mais parecidos com o XProlog pois possibilita relacionar tabelas de forma mais poderosa (como relações de uma para muitos e de muitos para muitos), mas não permite nem dedução, nem a utilização de uma abordagem recursiva para problemas de

recursividade, como é exemplificado mais a frente. Além disso é mais difícil de compreender do que o Power Query, requer algum tempo de interpretação e de prática, assim como também é fulcral a visualização de tutoriais introdutórios para se poder trabalhar com a ferramenta ao mais básico nível e um conhecimento mais profundo de forma a poder explorar todas as suas possibilidades.

Além disso, tal como referido já para o Power Query, a instalação deste *add-in* implica uma maior lentidão de arranque do Excel, maior que no Power Query e até mais lentidão durante a execução de tarefas no próprio Excel.

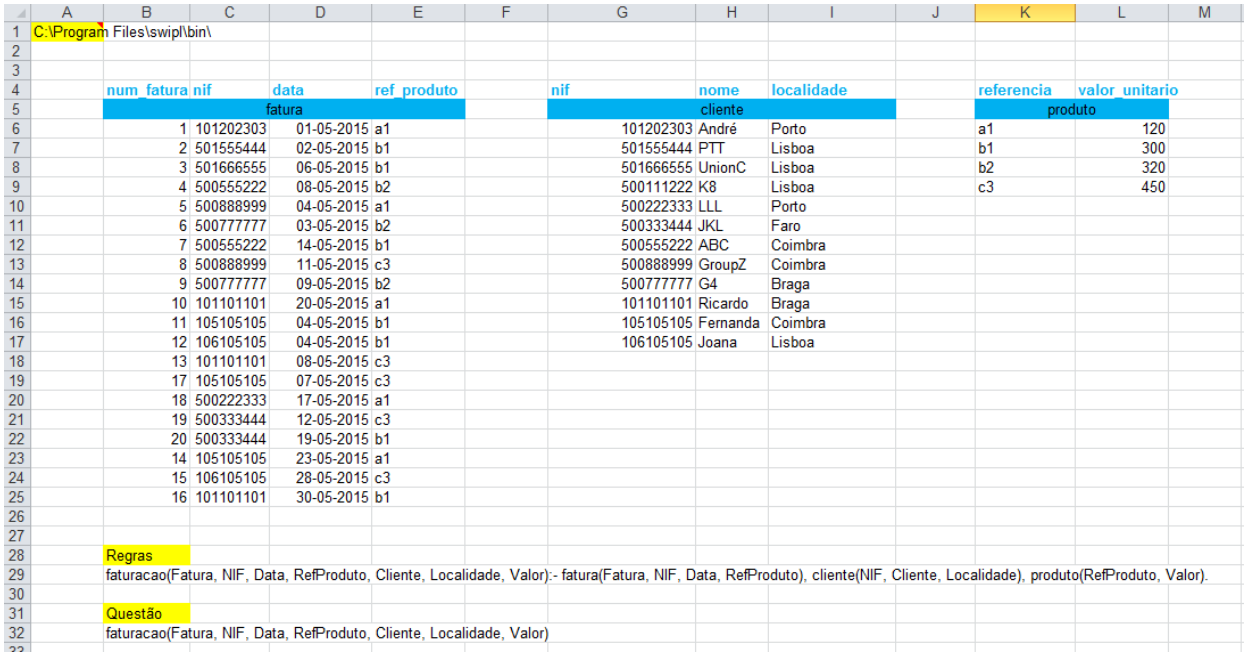
4.2.Exemplos e vantagens da utilização do XProlog

Foram criados e testados alguns exemplos em Excel tradicional (sem suplementos) e também comparados quando usado com Xprolog e Power Pivot. Através destes exemplos é possível perceber algumas das vantagens da utilização de folhas de cálculo dedutivas, mas essencialmente da utilização do Xprolog.

4.2.1 Exemplo com tabelas relacionais

Exemplo 1.1

O primeiro exemplo onde se verificam as vantagens do Xprolog em detrimento da folha de cálculo tradicional é na junção de dados de várias tabelas. O fato de se poder com apenas uma regra e uma questão em Prolog juntar em Excel, várias tabelas que tenham chaves de ligação entre si, ou seja, se na tabela A a coluna 1 corresponde à coluna 2 da tabela B, por exemplo, podemos rapidamente efetuar a ligação entre as tabelas referidas.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	C:\Program	Files\swipl\bin\											
2													
3													
4		num_fatura	nif	data	ref_produto		nif	nome	localidade		referencia	valor_unitario	
5		fatura					cliente				produto		
6		1	101202303	01-05-2015	a1		101202303	André	Porto		a1	120	
7		2	501555444	02-05-2015	b1		501555444	PTT	Lisboa		b1	300	
8		3	501666555	06-05-2015	b1		501666555	UnionC	Lisboa		b2	320	
9		4	500555222	08-05-2015	b2		500111222	K8	Lisboa		c3	450	
10		5	500888999	04-05-2015	a1		500222333	LLL	Porto				
11		6	500777777	03-05-2015	b2		500333444	JKL	Faro				
12		7	500555222	14-05-2015	b1		500555222	ABC	Coimbra				
13		8	500888999	11-05-2015	c3		500888999	GroupZ	Coimbra				
14		9	500777777	09-05-2015	b2		500777777	G4	Braga				
15		10	101101101	20-05-2015	a1		101101101	Ricardo	Braga				
16		11	105105105	04-05-2015	b1		105105105	Fernanda	Coimbra				
17		12	106105105	04-05-2015	b1		106105105	Joana	Lisboa				
18		13	101101101	08-05-2015	c3								
19		17	105105105	07-05-2015	c3								
20		18	500222333	17-05-2015	a1								
21		19	500333444	12-05-2015	c3								
22		20	500333444	19-05-2015	b1								
23		14	105105105	23-05-2015	a1								
24		15	106105105	28-05-2015	c3								
25		16	101101101	30-05-2015	b1								
26													
27													
28		Regras											
29		faturacao(Fatura, NIF, Data, RefProduto, Cliente, Localidade, Valor):- fatura(Fatura, NIF, Data, RefProduto), cliente(NIF, Cliente, Localidade), produto(RefProduto, Valor).											
30													
31		Questão											
32		faturacao(Fatura, NIF, Data, RefProduto, Cliente, Localidade, Valor)											

Figura 38 – Exemplo 1.1 em Xprolog.

No exemplo 1.1 temos o caso de três tabelas referentes a informação sobre a faturação de uma empresa e dados sobre os seus clientes e produtos. Na tabela "*fatura*" está a informação sobre o número da fatura, o número de contribuinte do cliente, a data do documento e a referência do produto vendido. Na tabela 2 designada "*cliente*", temos a informação sobre os clientes onde consta o NIF (número de identificação fiscal), o nome do cliente e a localidade de cada um. Por fim temos uma terceira tabela designada "produto" onde consta a referência e o valor do produto vendido. Utilizando a regra em Prolog que se verificar na figura 38, e colocando a questão também aí referida, obtemos o resultado da figura seguinte:

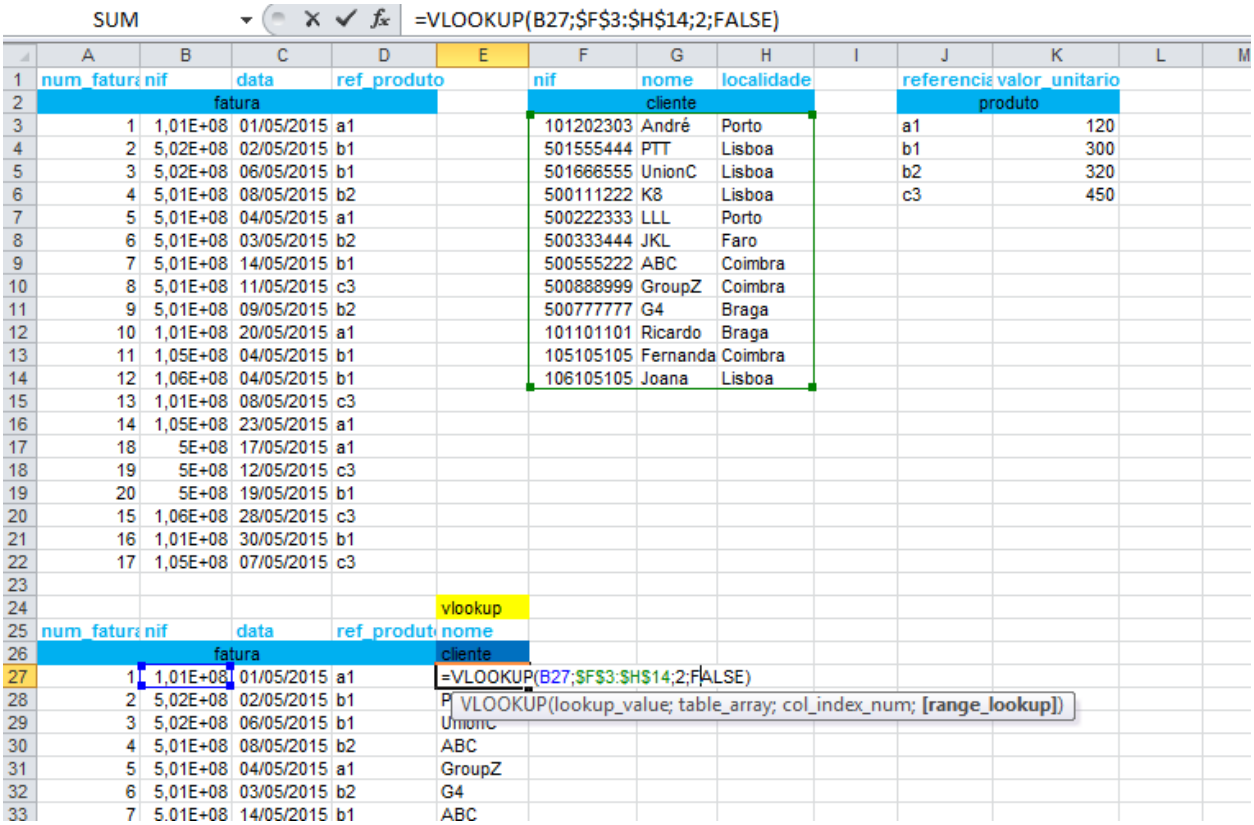
Output

Fatura	NIF	Data	RefProduto	Cliente	Localidade	Valor
1	101202303	05-01-2015	a1	andré	porto	120
2	501555444	05-02-2015	b1	ptt	lisboa	300
3	501666555	05-06-2015	b1	unionc	lisboa	300
4	500555222	05-08-2015	b2	abc	coimbra	320
5	500888999	05-04-2015	a1	groupz	coimbra	120
6	500777777	05-03-2015	b2	g4	braga	320
7	500555222	14-5-2015	b1	abc	coimbra	300
8	500888999	05-11-2015	c3	groupz	coimbra	450
9	500777777	05-09-2015	b2	g4	braga	320
10	101101101	20-5-2015	a1	ricardo	braga	120
11	105105105	05-04-2015	b1	fernanda	coimbra	300
12	106105105	05-04-2015	b1	joana	lisboa	300
13	101101101	05-08-2015	c3	ricardo	braga	450
17	105105105	05-07-2015	c3	fernanda	coimbra	450
18	500222333	17-5-2015	a1	lll	porto	120
19	500333444	05-12-2015	c3	jkl	faro	450
20	500333444	19-5-2015	b1	jkl	faro	300
14	105105105	23-5-2015	a1	fernanda	coimbra	120
15	106105105	28-5-2015	c3	joana	lisboa	450
16	101101101	30-5-2015	b1	ricardo	braga	300

Figura 39 – *Output* Xprolog.

Desta forma, conseguimos rapidamente unir três tabelas através das suas chaves de ligação, de uma forma rápida e relativamente simples.

Exemplificando em Excel tradicional com a utilização da fórmula *Vlookup* podemos verificar que os resultados seriam idênticos com o *Vlookup*, no entanto, verifica-se que a tarefa a realizar não é totalmente simples como se demonstra seguidamente, além de no Xprolog as regras usadas ficarem sempre visíveis, no Excel tradicional a fórmula usada fica escondida e as suas referências não são muitas vezes rapidamente perceptíveis e claras.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	num_fatura	nif	data	ref_produto		nif	nome	localidade		referencia	valor_unitario		
2	fatura					cliente				produto			
3	1	1,01E+08	01/05/2015	a1		101202303	André	Porto		a1	120		
4	2	5,02E+08	02/05/2015	b1		501555444	PTT	Lisboa		b1	300		
5	3	5,02E+08	06/05/2015	b1		501666555	UnionC	Lisboa		b2	320		
6	4	5,01E+08	08/05/2015	b2		500111222	K8	Lisboa		c3	450		
7	5	5,01E+08	04/05/2015	a1		500222333	LLL	Porto					
8	6	5,01E+08	03/05/2015	b2		500333444	JKL	Faro					
9	7	5,01E+08	14/05/2015	b1		500555222	ABC	Coimbra					
10	8	5,01E+08	11/05/2015	c3		500888999	GroupZ	Coimbra					
11	9	5,01E+08	09/05/2015	b2		500777777	G4	Braga					
12	10	1,01E+08	20/05/2015	a1		101101101	Ricardo	Braga					
13	11	1,05E+08	04/05/2015	b1		105105105	Fernanda	Coimbra					
14	12	1,06E+08	04/05/2015	b1		106105105	Joana	Lisboa					
15	13	1,01E+08	08/05/2015	c3									
16	14	1,05E+08	23/05/2015	a1									
17	18	5E+08	17/05/2015	a1									
18	19	5E+08	12/05/2015	c3									
19	20	5E+08	19/05/2015	b1									
20	15	1,06E+08	28/05/2015	c3									
21	16	1,01E+08	30/05/2015	b1									
22	17	1,05E+08	07/05/2015	c3									
23													
24					vlookup								
25	num_fatura	nif	data	ref_produto	nome								
26	fatura				cliente								
27	1	1,01E+08	01/05/2015	a1	=VLOOKUP(B27;$F$3:$H$14;2;FALSE)								
28	2	5,02E+08	02/05/2015	b1	P								
29	3	5,02E+08	06/05/2015	b1	UnionC								
30	4	5,01E+08	08/05/2015	b2	ABC								
31	5	5,01E+08	04/05/2015	a1	GroupZ								
32	6	5,01E+08	03/05/2015	b2	G4								
33	7	5,01E+08	14/05/2015	b1	ABC								

Figura 40 – Exercício 1.1. em Excel.

Como se pode verificar na figura 40, ao utilizar a função *Vlookup* consegue resolver este problema, mas para cada nova coluna que se queira juntar é necessário refazer a função de acordo com as referências pretendidas. Além disso o Xprolog quando dá resposta coloca o *output* como valores enquanto no Excel tradicional é necessário fazer *copy/paste* como '*values*' de forma a perder a fórmula agregada a cada célula do output. No Xprolog apesar das células ficarem como valores apenas e sem fórmula associada aparentemente, a célula superior direita do *output* fica sempre associada à base de conhecimento e à questão, pelo que qualquer alteração vai recalcular a resposta para todo o *output*.

Este exemplo podería ser resolvido também facilmente com o auxílio do Power Query. Uma das vantagens deste suplemento, como já referido, é a de poder importar para Excel dados de várias fontes e inclusive efetuar ligações entre as tabelas importadas, situação que solucionaria o problema referido acima rapidamente. No entanto existem as desvantagens, sobretudo iniciais, para os utilizadores deste suplemento, tais como a necessidade de observar tutoriais iniciais para se conseguir dominar a ferramenta, assim como a necessidade da sua instalação e software

complementar (exemplo do .NET Framework), além de que são ferramentas mais pesadas do que o Xprolog.

Exemplo 1.2

Imaginado que temos uma sequência deste exemplo, imagine-se que se pretende inicialmente, por exemplo, não apenas juntar as tabelas mas obter o total de vendas por localidade, ou numa localidade em específico. Para o fazer mais rapidamente do que em Excel tradicional pode ser usado o suplemento Power Pivot para facilitar a tarefa. Em Xprolog, a utilização de código Prolog na questão é suficiente para se poder obter rapidamente a resposta pretendida. Como se pode verificar na figura seguinte:

Exemplo 1.2 em Xprolog:

Reaproveitando a resposta à primeira questão colocada, para responder à nova pergunta "Total de vendas com cliente de Coimbra?"

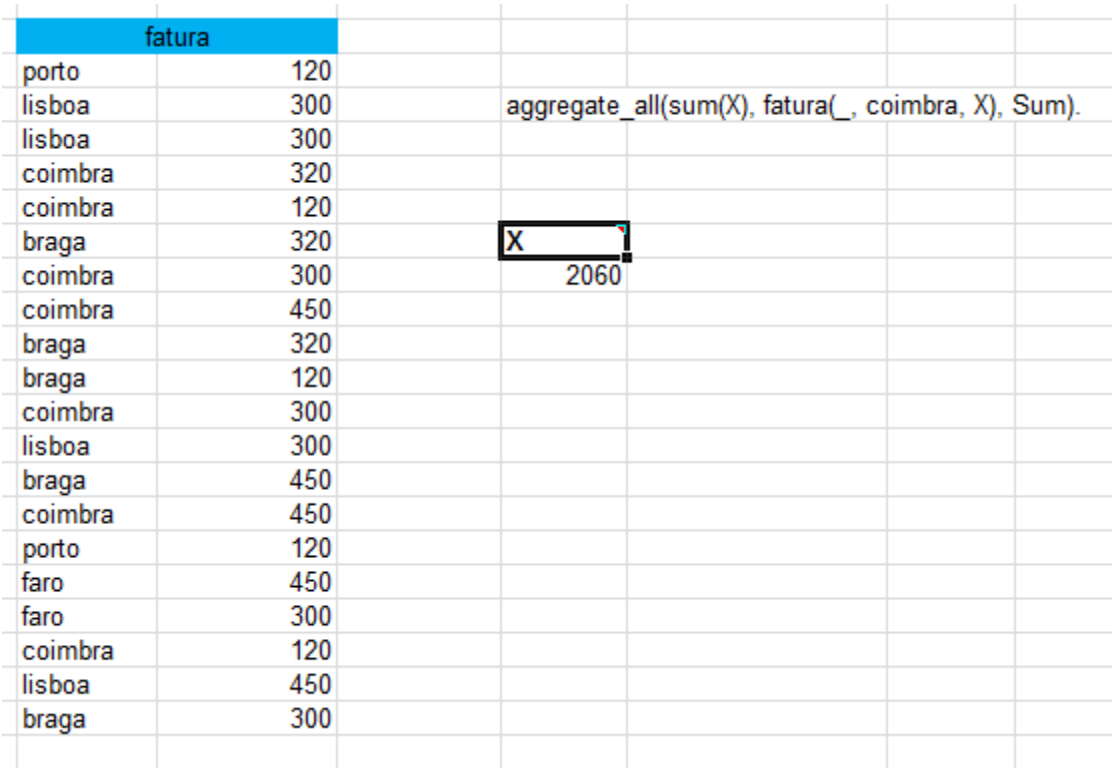
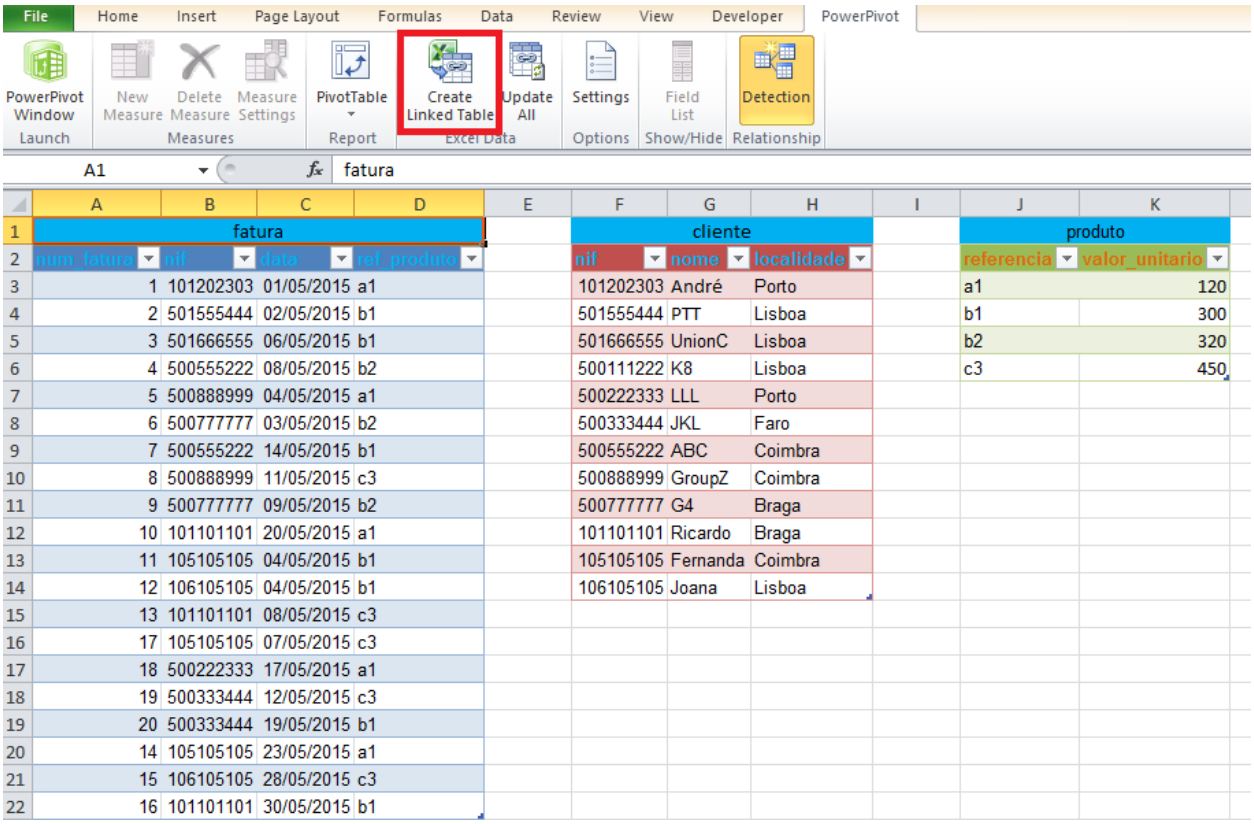


Figura 41 – Exemplo 1.2 em Xprolog.

Uma questão em Prolog como a exemplificada acima dá imediatamente a resposta à questão colocada.

Exemplo 1.2 com Power Pivot:

Como referido anteriormente esta tarefa também é bastante mais simples com a utilização do Power Pivot, no entanto, a sua utilização exige um conhecimento mais avançado em Excel, mas sobretudo um conhecimento sobre o funcionamento deste suplemento o que em termos práticos se torna mais complexo e demorado para um utilizador inicial. Apresentamos seguidamente um exemplo de resolução com o auxílio do Power Pivot.



	A	B	C	D	E	F	G	H	I	J	K
1	fatura					cliente				produto	
2	num_fatura	nif	data	ref_produto		nif	nome	localidade		referencia	valor_unitario
3	1	101202303	01/05/2015	a1		101202303	André	Porto		a1	120
4	2	501555444	02/05/2015	b1		501555444	PTT	Lisboa		b1	300
5	3	501666555	06/05/2015	b1		501666555	UnionC	Lisboa		b2	320
6	4	500555222	08/05/2015	b2		500111222	K8	Lisboa		c3	450
7	5	500888999	04/05/2015	a1		500222333	LLL	Porto			
8	6	500777777	03/05/2015	b2		500333444	JKL	Faro			
9	7	500555222	14/05/2015	b1		500555222	ABC	Coimbra			
10	8	500888999	11/05/2015	c3		500888999	GroupZ	Coimbra			
11	9	500777777	09/05/2015	b2		500777777	G4	Braga			
12	10	101101101	20/05/2015	a1		101101101	Ricardo	Braga			
13	11	105105105	04/05/2015	b1		105105105	Fernanda	Coimbra			
14	12	106105105	04/05/2015	b1		106105105	Joana	Lisboa			
15	13	101101101	08/05/2015	c3							
16	17	105105105	07/05/2015	c3							
17	18	500222333	17/05/2015	a1							
18	19	500333444	12/05/2015	c3							
19	20	500333444	19/05/2015	b1							
20	14	105105105	23/05/2015	a1							
21	15	106105105	28/05/2015	c3							
22	16	101101101	30/05/2015	b1							

Figura 42 – Exemplo com Power Pivot.

Como podemos ver na figura 42 é necessário configurar as tabelas criadas como tabelas de Excel e depois no tabulador do Power Pivot selecionar "*Create Linked Table*".

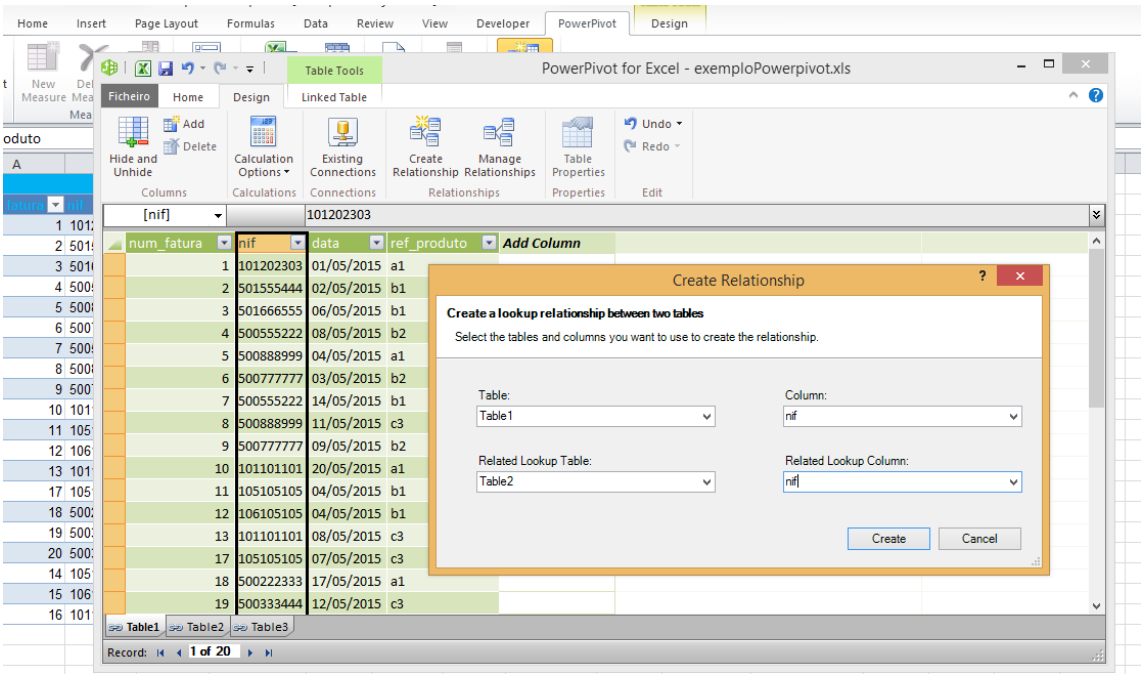


Figura 43 – Exemplo em Power Pivot.

De seguida é necessário criar as ligações entre as tabelas e as respetivas colunas, como se pode verificar na figura 43. Na figura 44 verifica-se a criação da segunda ligação entre a tabela 1 e a tabela 3.

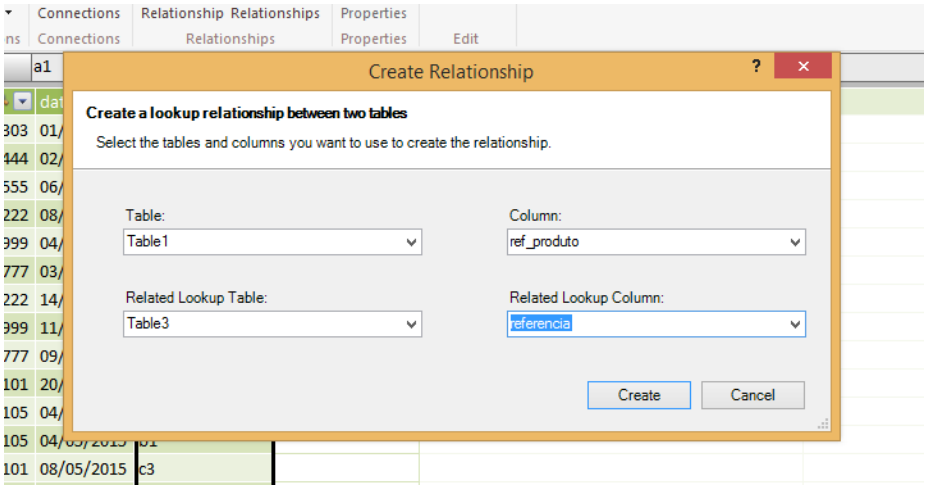


Figura 44 – Exemplo em Power Pivot.

Por fim é necessário selecionar a o botão da "*pivot table*" e escolher os campos a utilizar e a sua colocação na tabela dinâmica de forma a se obter a resposta à questão "Total de vendas com cliente de Coimbra?" que serão 2060 unidades neste caso.

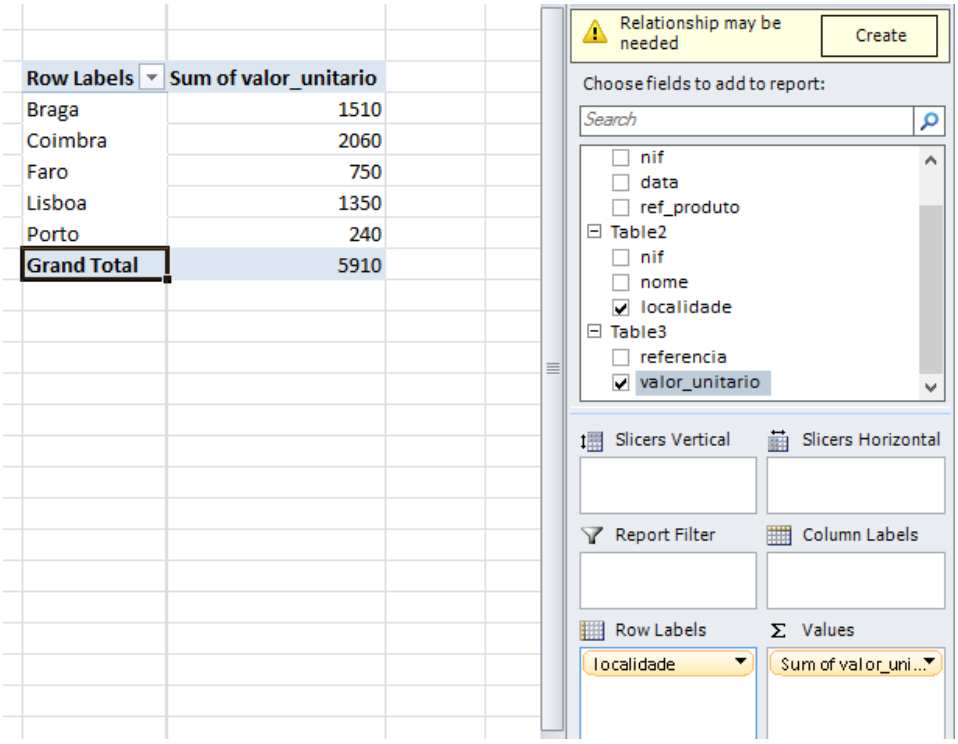


Figura 45 – Exemplo em Power Pivot.

4.2.2 Exemplo com recursividade

Um exemplo muito simples mas que em Excel é bastante complicado e por vezes poderá ser mesmo impossível de se resolver é uma situação que envolve recursividade. Na figura seguinte podemos ver uma base de dados onde estão os voos realizados por uma companhia aérea, e os dias da semana em que opera.

Origem	Destino	Dia
	voo	
Porto	Lisboa	Segunda
Lisboa	Porto	Segunda
Porto	Lisboa	Quarta
Lisboa	Porto	Quarta
Porto	Lisboa	Sexta
Lisboa	Porto	Sexta
Porto	Lisboa	Domingo
Lisboa	Porto	Domingo
Lisboa	Istambul	Terça
Istambul	Lisboa	Terça
Lisboa	Istambul	Sábado
Istambul	Lisboa	Sábado
Porto	Madrid	Terça
Madrid	Porto	Terça
Porto	Madrid	Quarta
Madrid	Porto	Quarta
Porto	Madrid	Quinta
Madrid	Porto	Quinta
Porto	Madrid	Sexta
Madrid	Porto	Sexta

regras

ligacao(Origem,Destino):-voo(Origem,Destino,Dia).

ligacao(Origem,Destino):-voo(Origem,X,Dia),ligacao(X,Destino).

questão 1

ligacao(lisboa,porto)

Output

yes

Figura 46 – Base de conhecimento em Xprolog.

As regras são definidas de forma a que no caso de não haver um voo direto de uma cidade para outra, das existentes na base de dados, o Xprolog possa saber se existem ligações indiretas, ou seja com uma ou mais escalas, ou pontos intermédios.

No primeiro exemplo, podemos verificar que para a questão "*existe ligação entre Lisboa e Porto?*" , o Xprolog verifica os voos com origem em "*Lisboa*", depois desses, os que têm destino "*Porto*" e vendo que existe a ligação dá resposta afirmativa.

Na questão 2 da figura 47 é demonstrado um exemplo mais explícito de uso de recursividade. O Xprolog vai verificar a existência de uma ligação indireta. Ou seja, é colocada a questão "Existe uma ligação com origem no Porto e destino Istambul ?" O Xprolog vai verificar que não existe ligação direta, mas vai encontrar ligação desde Porto a Lisboa e desde Lisboa a Istambul e por recursividade vai responder afirmativamente ao utilizador.

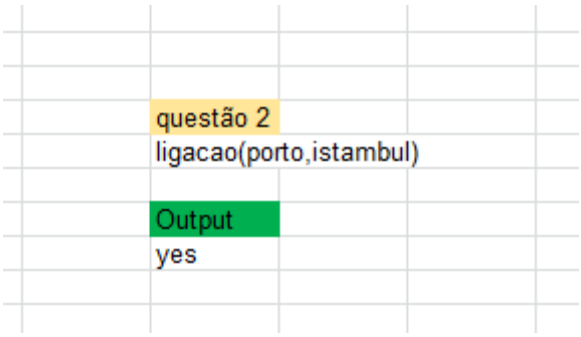


Figura 47 – Exemplo em Xprolog.

Em caso de utilização da folha de cálculo tradicional, para a primeira questão o problema também se consegue resolver, mas de uma forma não tão direta como com o Xprolog, inicialmente colocam-se filtros na tabela da base de dados, e começa-se por pesquisar na Origem "*Lisboa*" e depois no Destino "*Porto*" como se verifica na figura seguinte.

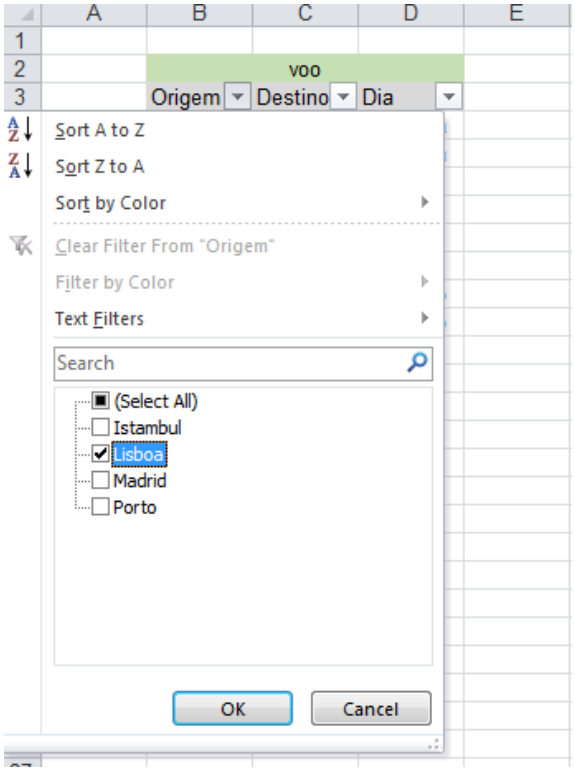


Figura 48 – Exemplo em Excel.

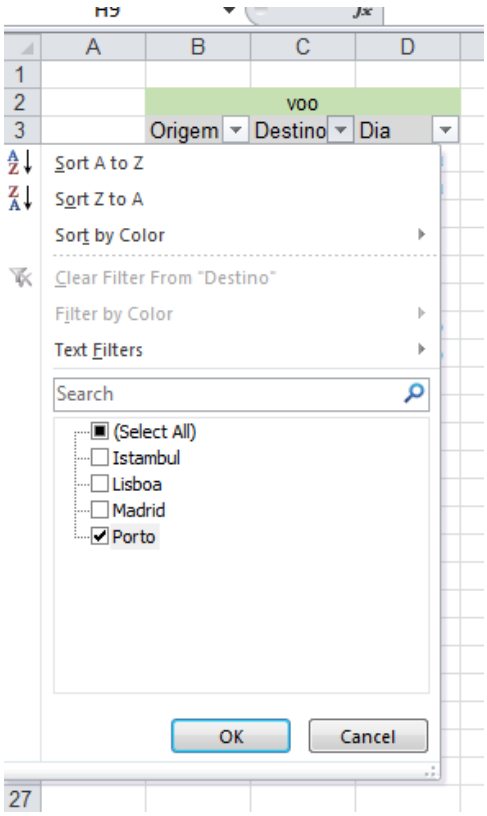


Figura 49 – Exemplo em Excel.

Neste caso o resultado final seria o verificado na figura 50.

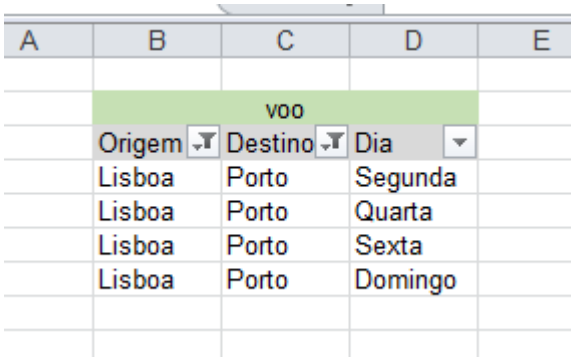
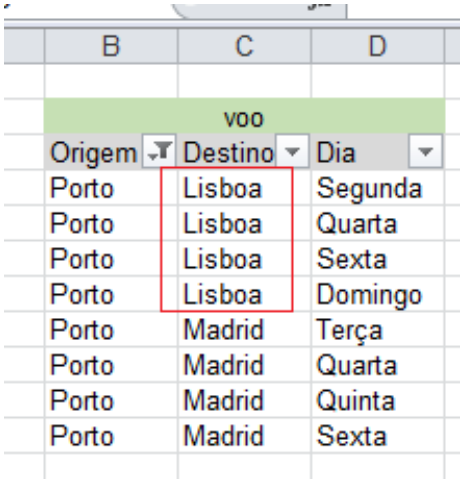


Figura 50 – Exemplo em Excel.

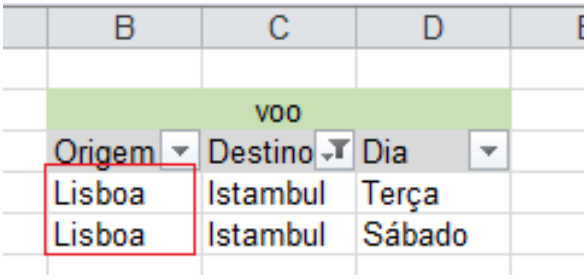
Agora, a questão 2 colocada é que seria bem mais complicada. Imaginando por exemplo, uma companhia aérea com centenas de ligações aéreas, tendo-as numa base de dados carregada em

Excel tradicional, nos moldes apresentados no exemplo anterior, encontrar a resposta à questão 2 poderia demorar imenso tempo e teria de ser feito em vários passos.



	B	C	D
		voo	
	Origem	Destino	Dia
	Porto	Lisboa	Segunda
	Porto	Lisboa	Quarta
	Porto	Lisboa	Sexta
	Porto	Lisboa	Domingo
	Porto	Madrid	Terça
	Porto	Madrid	Quarta
	Porto	Madrid	Quinta
	Porto	Madrid	Sexta

Figura 51 – Exemplo em Excel.



	B	C	D
		voo	
	Origem	Destino	Dia
	Lisboa	Istambul	Terça
	Lisboa	Istambul	Sábado

Figura 52 – Exemplo em Excel.

Como se pode verificar nas figuras anteriores, a forma de se conseguir perceber uma possível ligação seria sempre com uma análise minuciosa, inicialmente com todos os destinos desde Origem "Porto" e depois todas as Origens com destino "Istambul". E dos resultados apresentados perceber qual o elo de ligação. Neste caso seria através de Lisboa.

Além deste tipo de informação que se consegue obter, pode-se ainda questionar através do Xprolog quais as cidades intermédias ou escalas entre duas cidades que se pretende ligação, além de se poder questionar sobre quais os dias das respetivas ligações ou outro argumento, como o tipo de transporte a utilizar, mediante a utilização de regras definidas em Prolog e respetivas interrogações, como se pode verificar nos exemplos das figuras 53 e 54.

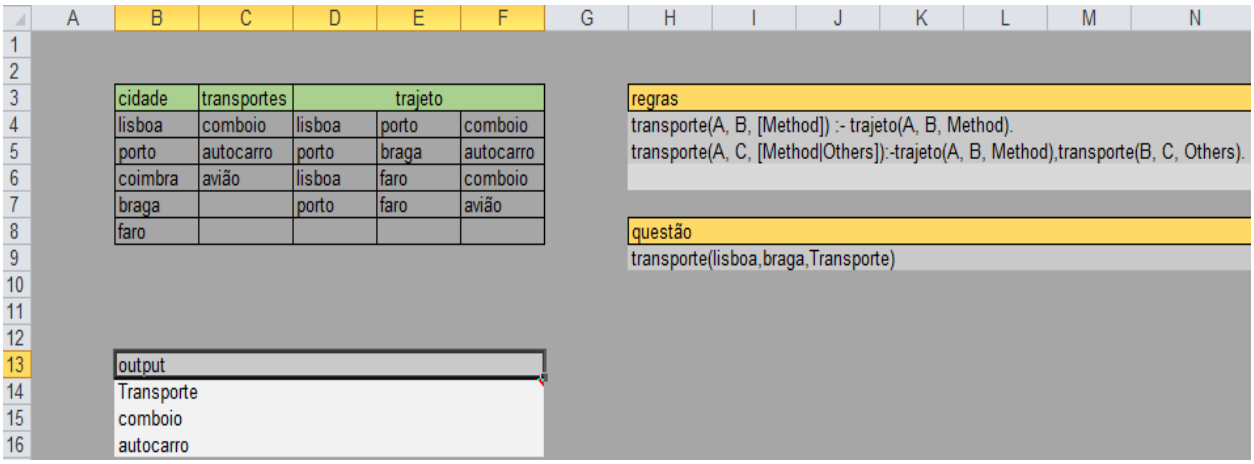


Figura 53 – Exemplo em Xprolog.

Nestes exemplos verificam-se resultados de situações que seriam bastante mais complicadas de obter em Excel tradicional, tendo ainda a vantagem, de podendo alterar a base de dados ou a questão, produzir alterações no Output, pelo que as regras definidas podem ser reaproveitadas para situações semelhantes com bases de dados.

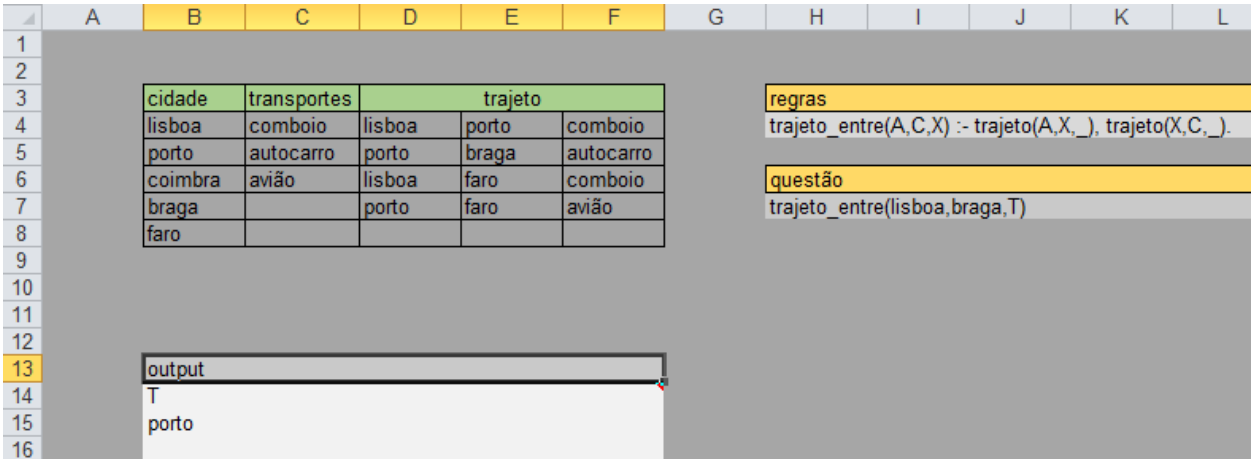


Figura 54 – Exemplo em Xprolog.

4.2.3.Exemplo de relações entre tabelas do tipo “muitos para um”

No exemplo da figura 55, temos uma base de dados com informação de um café e as primeiras quatro faturas do início deste ano. Para cada fatura, pode haver mais do que um item que representam diferentes produtos comprados pelos clientes do café, ou seja, existe uma relação de “muitos produtos para uma fatura” como referido no título do exemplo.

	A	B	C	D	E	F
1						
2		Nº	Item	Produto	Valor	
3		Fatura				
4		2015/01	1	Café	1	
5		2015/01	2	Chá	1	
6		2015/02	1	Capuccino	2	
7		2015/03	1	Capuccino	2	
8		2015/03	2	Água	1	
9		2015/03	3	Bolo	1	
10		2015/04	1	Café	1	
11		2015/04	2	Bolo	1	
12						
13						

Figura 55 – Exemplo de relações entre tabelas.

Imaginemos que o gerente do café tem esta informação carregada em Excel, e pretende saber quantas faturas não têm café ou água nos seus itens (produtos). Para analisar esta questão, uma das alternativas usando Excel tradicional seria utilizando esta fórmula:

=IF(AND(COUNTIFS($A$3:$A$10;$A3;$C$3:$C$10;"Café")=0;COUNTIFS($A$3:$A$10;$A3;$C$3:$C$10;"Água")=0);IF(COUNTIFS($A$2:$A2;A2;$E$2:$E2;"Ok")=0;"Sim";"");"")

Temos de aplicar esta fórmula complexa a todas as células de uma nova coluna anexada à tabela principal como exemplificado na figura 56.

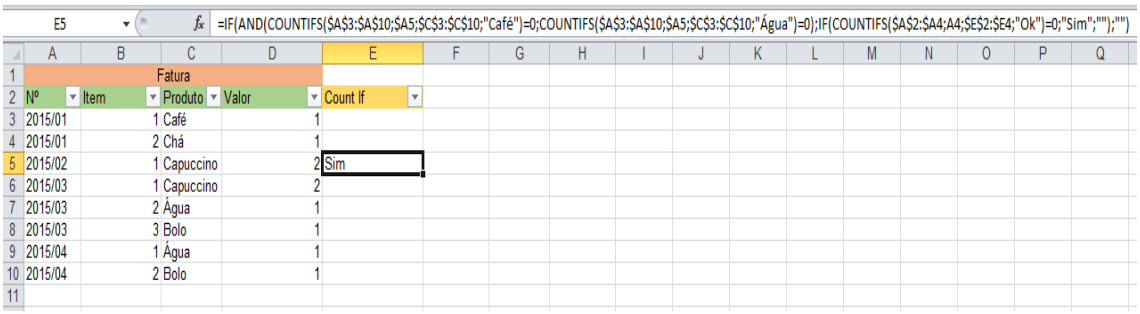


Figura 56 – Resolução em Excel.

Por fim, é necessário filtrar pela coluna nova adicionada (neste caso "*Count If*") todas as respostas positivas ("*Sim*"). O resultado final será o representado na figura abaixo:

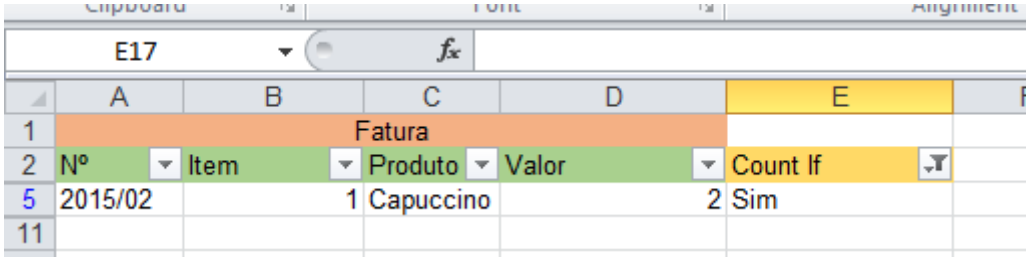


Figura 57 – Resolução em Excel.

Usando o Xprolog, coloca-se a questão em Prolog na figura 58.

O *output* apresentado mostra-nos o número de faturas nos requisitos pretendidos, assim como nos indica a ou as faturas nessas condições.

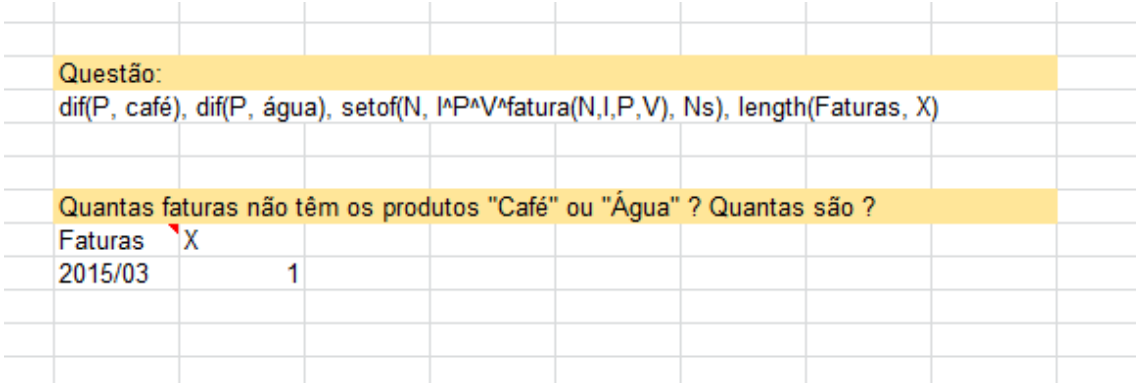


Figura 58 – Resolução em Xprolog.

Capítulo V - Conclusão

5.1.Desenvolvimentos futuros no XProlog

Tendo em conta as limitações apontadas no capítulo IV, existe possibilidades de desenvolvimentos futuros que visem aumentar a capacidade já existente do Xprolog, assim como adicionar funcionalidades novas que melhorem não só a utilidade, mas também a performance da ferramenta desenvolvida.

Uma melhoria do sistema seria a liberdade de definir várias áreas como base de conhecimento, não tendo as mesmas de ser contíguas. Isto vai permitir ao utilizador aproveitar por exemplo uma área de *output* como parte de uma nova base de conhecimento, mantendo a anterior para a mesma questão a colocar.

Outra situação a resolver é a colocação de "*roles*" nas tabelas de base de dados, permitindo que para determinado predicado, os seus argumentos tenham uma designação genérica, como é apontado no ponto 3.4 deste artigo. Esta é uma situação que à partida não será de difícil resolução, mas ainda não se conseguiu implementar. Como se pode verificar na figura 59, o carregamento de dados torna-se mais simples e compreensível, com a primeira linha a corresponder ao nome do predicado "Fatura" e a segunda à designação genérica dos seus argumentos "Numero_Fatura", "Nome_Cliente" e "Produto".

Fatura		
Numero_Fatura	Nome_Cliente	Produto
1001	José	Batatas
1002	Marco	Cebolas
1003	António	Cebolas
1004	Eva	Batatas
1005	Maria	Cebolas
1006	José	Alho
1007	Marco	Batatas

Figura 59 - Possível desenvolvimento futuro da base de conhecimento no XProlog.

Relativamente à sintaxe da ferramenta, a melhoria que se perspetiva é tornar mais simples e mais fácil de compreensão pelos utilizadores, ou seja, fazer com que mais facilmente um utilizador básico de Excel, mas sobretudo com poucos ou praticamente nenhuns conhecimentos de Prolog, possa rapidamente trabalhar com a ferramenta de forma independente. O objetivo futuro passa por permitir a utilizadores sem conhecimento em programação lógica conseguirem interativamente definir regras e definir interrogações sobre a base de conhecimento de forma tão simples como são definidas as fórmulas em Excel. Apesar da motivação para ultrapassar esta limitação, esta situação não se afigura uma tarefa simples nem de rápida execução.

Existem ainda várias outras características da ferramenta que poderão ser estendidas, como por exemplo o seu interface ser mais apelativo aos utilizadores, ou desenvolver capacidades extra-Prolog que permitam, por exemplo usar os efeitos de multi-direcionalidade identificados e referidos noutras folhas de cálculo dedutivas, tendo sempre em consideração que o objetivo inicial e pelo qual continuamos a trabalhar será o de desenvolver folhas de cálculo com capacidades de representação de conhecimento e de raciocínio.

5.2.Considerações Finais

O objetivo deste trabalho é o de estender as capacidades da folha de cálculo tradicional, desenvolvendo uma folha com capacidades de representação de conhecimento e de raciocínio. A ideia de ferramenta ideal seria aquela onde o utilizador final conseguisse resolver o mais facilmente problemas complexos e que numa tradicional folha de cálculo seria bastante complicado e demorado ou mesmo impossível de resolver. Tendo em conta que conseguindo efetuar a ligação entre o Excel e o interpretador Prolog, o SWI-Prolog, muitas destas situações ficariam mais simplificadas, uma boa parte do que se pretende está já realizado.

A ferramenta que apresentamos nesta dissertação é o Xprolog, que apesar de não ser ainda um produto final que possa ser usado pelo utilizador, permite já nos desenvolvimentos adquiridos com este sistema, efetuar tarefas e obter resultados numa folha de cálculo tradicional que normalmente são bastante complexos e difíceis ou mesmo impossíveis de obter sem ajuda de suplementos (add-ins). Pareceu-nos ainda importante a realização deste trabalho devido à falta de ferramentas que possam ser consideradas como soluções alternativas ao Xprolog. Apesar das várias folhas de cálculo dedutivas estudadas e apresentadas resumidamente, a maior parte delas não estão disponibilizadas ao público em geral e mesmo as que possam estar são de difícil acesso ou não são disponibilizadas gratuitamente na sua totalidade.

Na apresentação do Xprolog, tentou-se mostrar com exemplos claros e comparativos as vantagens da utilização não só da ferramenta em si, mas de um tipo de aplicação que é bastante útil, apesar de pouco explorado pela comunidade informática. Os desenvolvimentos futuros serão de extrema importância, por um lado para dar continuidade à ferramenta criada, por outro para continuar a promover a atenção da comunidade informática, e em especial da comunidade de inteligência artificial, neste tipo de aplicações e na sua importância e mais valia para as suas causas e objetivos.

Outra das metas que se procuram alcançar com este trabalho é a de dar um novo ênfase a uma temática que nos últimos tempos parece ter caído em esquecimento, ou pelo menos, ter decaído no grande interesse da comunidade informática em geral. Sendo esta uma tecnologia que pode ser bastante útil na área de inteligência artificial, não tem prevalecido e assegurado a sua vital importância pois não se tem conseguido criar as ferramentas úteis e fáceis de manejar.

Capítulo VI - Bibliografia

Bratko, I. (2001). *Prolog Programming for Artificial Intelligence, 3rd edition*. Pearson Education / Addison-Wesley.

Bringsjord, S. (1997). *Historical roots of logic programming (LP).* Web Site: http://homepages.rpi.edu/~brings/LOG+AI/lai/node15.html

Casanova, M.A., Giorno, F.A.C. & Furtado, A.L. (1987). *Programação em Lógica e a Linguagem Prolog*. IBM Brasil, Centro Científico do Rio de Janeiro. Editora Edgard Blücher, Ltda.

Cervesato, I. (2005). *The deductive spreadsheet,* Technical Report DS05-02, Deductive Solutions.

Cervesato, I. (2007). *NEXCEL, a Deductive Spreadsheet.* Computer Science Department. Carnegie Mellon University.

Chalupsky, H., Macgregor, R.M., Russ, T. (2006). *PowerLoom Manual.* ISI, Marina Del Rey, CA.

Covington, M. A., Nute, D. & Vellino, A. (1997). *Prolog Programming in Depth.* Prentice-Hall.

Dodd, T. (1990). *Prolog: A Logical Approach.* New York: Oxford University Press.

Forgy, C.L. (1974). *A network match routine for production systems*, Working Paper.

Grad, B. (2007). *The Creation and Demise of Visicalc.* IEEE Annals of History of Computing 29(3):20-31.

Gunning, D. (2004). DARPA SBIR 2004.3, topic sb043-040: *Deductive spreadsheets*.

Gupta, G. & Akhter, S. (2000). *Knowledgesheet: A graphical spreadsheet interface for interactively developing a class of constraint programs,* in 'Proceedings of the Second International Workshop Practical Aspects of Declarative Languages', Springer Verlag LNCS 1753, Boston, MA.

Hogger., C. J. (1990). *Essentials of Logic Programming.* College of Science and Technology, London, UK.

Huang, S. S., Green, T. J., Loo, B. T. (2011). *Datalog and recursive query processing. Foundations and Trends in Databases.* In preparation.

Kassoff, M., Zen, L.-M., Garg, A. & Genesereth, M. (2005). *Predicalc: A logical spreadsheet management system*. In 31st International Conference on Very Large Databases (VLDB).

Kassoff, M. & Valente, A. (2007). An introduction to logical spreadsheets. *Knowl. Eng. Rev.*, 22(3), 213-219.

Kriwaszek, F. (1988). LogiCalc – A Prolog Spreadsheet. in D. Michie and J. Hayes, *Machine Intelligence* 11.

Oliveira, G. S. & Silva, A. F. (2013). Prolog: A Linguagem, A Máquina Abstrata de Warren e Implementações. *Revista de Informática Teórica e Aplicada: RITA* , v. 20, p. 214-246.

Palazzo, L. A. M. (1997). *Introdução à Programação Prolog.* Pelotas: EDUCAT - Editora da Universidade Católica de Pelotas.

Power, D. J. (2004). *A Brief History of Spreadsheets*, DSSResources.COM. Web Site: http://dssresources.com/history/sshistory.html

Ramakrishnan, C. R., Ramakrishnan, I. V., & Warren, D. S. (2007). XcelLog: A deductive spreadsheet system. *The Knowledge Engineering Review*, 22(03), 269–279.

Salomon, D. (1991). *Como fazer uma monografia.* São Paulo. Martins Fontes.

Scoville, R. (1999). *PC-Bible*, Chapter 15: Spreadsheets, p. 403.

Spenke, M. & Beilken, C. (1989). *A spreadsheet interface for logic programming*, in 'CHI '89: Proceedings of the SIGCHI conference on Human Factors in Computing Systems', ACM Press, pp. 75–80.

Sterling, L. & Shapiro, E.Y. (1986). *The Art of Prolog: Advanced Programming Techniques.* The MIT Press.

Tallis, M., Waltzman, R., & Balzer, R. (2007). Adding deductive logic to a COTS spreadsheet. *The Knowledge Engineering Review*, 22(03), 255–268.

Tukiainen, M. (2001). *Developing a New Model of Spreadsheet Calculation: A Goals and Plans Approach.* University of Joensuu.

Valente, A., van Brackle, D., Chalupsky, H., & Edwards, G. (2007). Implementing logic spreadsheets in LESS. *The Knowledge Engineering Review*, 22(03), 237–253.

van Emden, M. H., Ohki, M. & Takeuchl, A. (1986) Spreadsheets with incremental queries as a user interface for logic programming. *New Generation Comput.* 4, 287-304.

van Emden, M. H., Cheng, M. H. M. & Lee, J. H. M. (1988). *Tables as user interfaces for logic programs.* Int. Conv. 5th Generation Computer Systems, Tokyo.

Warren, D. S. & Swift, T. (1993). *A Summary of XSB Performance.*

Wielemaker, J. (Updated for version 3.2.9, July 1999). *SWI-Prolog 3.2 - Reference Manual*. University of Amsterdam.

Xavier, G. F. C. (2007). *Lógica de Programação.* São Paulo. SENAC, São Paulo.